AN ADAPTIVE MODEL FOR MULTI-MODAL BIOMETRICS DECISION FUSION

TRAN QUOC LONG

NATIONAL UNIVERSITY OF SINGAPORE
2005

AN ADAPTIVE MODEL FOR MULTI-MODAL BIOMETRICS DECISION FUSION

TRAN QUOC LONG
(B.Sc, Vietnam National University)

A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2005

Name: Tran Quoc Long
Degree: Master of Engineering
Department: Electrical and Computer Engineering
Title: An Adaptive Model for Multi-Modal Biometrics Decision Fusion

Abstract

Multi-modal biometric verification is gaining more and more attention recently because of the high security level it provides and the non-universality of uni-modal biometrics. Multi-modal biometrics decision fusion can be considered as a classification task since the output is either a genuine user or an impostor. This treatment allows many available classifiers to be applied in the field. In this thesis, two problems related to multi-modal biometrics decision fusion are considered. The first problem is new user registration. Frequent registration not only requires storing of new patterns into the biometric database but also requires updating the combination module efficiently. The second problem is related to sensor decay which results in change of matching scores with time. The performance of a fixed classifier may be affected for such case. In this thesis, an adaptive algorithm to solve these problems has been proposed. This algorithm can update the combination module whenever new training patterns are available without having to retrain the module from scratch. The new algorithm is demonstrated using experiments on physical application data to address both the registration and matching scores distribution changing problems using three biometrics, namely fingerprint, speech and hand-geometry.

Keywords: Multi-modal biometrics, decision fusion, biometrics verification, recursive least squares, parameter estimation.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors, Dr. Kar-Ann Toh and professor Dipti Srinivasan, for their constant support, guidance and motivation. I take this opportunity to thank the staff members in the Department of Electrical and Computer Engineering for helping me with the administrative details regarding my thesis. I would also like to thank the Institute for Infocomm Research, Singapore for sponsoring my research scholarship.

In addition, I would like to thank my friends at the institute, Pham Nam Trung, Pham Duc Minh and Phung Minh Hoang, for many helpful discussions we had together and my roommates, Dinh Trung Hoang, Alin Chitu and Jonathan Stern, for their advices on many stubborn problems in academics as well as in life.

I would also like to thank my parents, my brother and the rest of my family members for their endless support to all my endeavors. Finally, I would like to express my deepest gratitude to Tran Thuy Anh for all the encouragement, support and fun she has provided throughout my stay in Singapore.

Contents

Summary

Multi-modal biometric verification is gaining more and more attention recently because of the high security level it provides and the non-universality of uni-modal biometrics. Multi-modal biometrics decision fusion can be considered as a classification task since the output is either a genuine user or an impostor. This treatment allows many available classifiers to be applied in the field. In this thesis, two problems related to multi-modal biometrics decision fusion have been considered. The first problem is new user registration. Frequent registration not only requires storing of new patterns into the biometric database but also requires updating the combination module efficiently. The second problem is related to sensor decay which results in change of matching scores with time, thereby affecting the performance of a fixed classifier.

In order to choose a suitable classifier for multi-modal biometrics decision fusion, extensive empirical comparison of several classifiers using real world data sets was conducted in this research. These experiments focussed on classifier training time, memory storage requirements, and classification accuracy. The experimental results are reported in detail along with a discussion on selecting a suitable classifier as a basis for an efficient multi-modal biometric verification system.

After carefully selecting a suitable classifier, main focus of this thesis is the development of an adaptive algorithm for multi-modal biometrics decision fusion. This adaptive algorithm has been proposed to solve the registration and sensor decay problems mentioned above. The algorithm can update the combination module whenever new training patterns are available without having to retrain the module all over from scratch.

Finally, the new algorithm was evaluated using experiments on physical application data to address both the registration and sensor decay problems. Temporal biometric data sets for a reasonably long period were collected for this evaluation. The data sets consist of three biometrics, namely fingerprint, speech and hand-geometry. The

experimental results showed that the new algorithm is superior to the original algorithm in the registration process and when there are changes in matching scores with time.

List of Figures

List of Tables

Publications related to this thesis

1. Kar-Ann Toh, Quoc-Long Tran, and Dipti Srinivasan. "Benchmarking a Reduced Multivariate Polynomial Pattern Classifier". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6): pp. 740–755, 2004.

2. Quoc-Long Tran, Kar-Ann Toh, and Dipti Srinivasan. "Adaptation to Changes in Multimodal Biometric Authentication". *1st IEEE Conference on Cybernetics and Intelligent Systems (CIS'04)*, pp. 981–985, 2004.

3. Kar-Ann Toh, Quoc-Long Tran, and W.-Y. Yau. "Some Learning Issues Pertaining to Adaptive Multimodal Biometric Authentication". Proceedings of the Fifth Chinese Conference on Biometrics Recognition (LNCS 3338), pp. 617–628, 2004 (invited paper).

4. Quoc-Long Tran, Kar-Ann Toh, and Dipti Srinivasan. "An Empirical Comparison of Nine Pattern Classifiers". *IEEE Transactions on Systems, Man and Cybernetics, Part B*, to appear in August, 2005.

Chapter 1

Introduction

Security systems are widely implemented in office buildings to prevent fraudulent access. These security systems can be either manual or automatic. In both cases, such systems must rely on certain means to identify or verify human beings. The study on the use of such means is central to the development and implementation of an efficient a security system.

Let us begin with a typical example. John is an employee working in the Singapore Airport. Every morning, when he goes to office, the security guard asks him ”Good morning, please show your badge”. John says ‘hi’ to him and shows his badge. After checking, the guard let John go inside the building. This short conversation happens everyday in office buildings. In fact, similar schemes appear in other activities like security system access, business transactions, and etc. All are related to a common security issue - identification or verification of human beings.

Let us examine the example a little more. When the guard asks John to show his badge, in fact, he asks John to show him some proof that John is an inside person and has the right to go in. If John shows him the correct badge which is the proof, he is allowed to go in, otherwise, he is not allowed. One may argue that, sometimes the security guard, having known John for a long time, lets him go in without asking for the badge. So where is the proof? Strictly speaking, John’s face is his proof. The guard

recognizes John as an inside person through his face and lets him in.

The above example shows that identification and verification of human beings relate to some kind of proof. The proof dictates that a certain human being has the right to access the system or to do some specific job. So far, many kinds of such proof have been developed [78]. For example, identity cards, passwords, personal identification numbers (PIN) are very common. Recently, human physical and behavioral characteristics, such as fingerprint, face, speech, signature, and etc. have been utilized for automatic identification and verification purposes. These characteristics are called *human biometrics* [24]. This research focusses on use of multiple biometrics in verification of human beings.

1.1 Need for Biometric Verification

The classical verification techniques based on "what you have" or "what you know" like ID cards, passwords, PINs have many drawbacks [24, 78]. Passwords and PIN can be forgotten and uncovered due to users' carelessness. Identity cards can be lost or stolen. Strictly speaking, these techniques cannot truly help the system to distinguish a registered user from an impostor because they give authority to the ID cards, passwords or PIN, but not to the user himself. Anyone who has the cards or passwords is given the right to access. Thus, stolen cards, passwords or PIN raise a serious problem especially in highly secure systems and in business transactions. Another discomfort when utilizing these techniques is that people have to remember tens of passwords and PINs, and store tens of cards in their pocket for different security systems.

Perhaps, biometric is the most promising type of proof that can circumvent the problems mentioned above. Biometric identification is based on human physical or behavioral characteristics (i.e. "what you are") which are believed to be unique for each person. Because of this uniqueness, biometric identification and verification systems are less prone to fraud. Also, human biometrics such as fingerprint and speech

are difficult to be lost or forgotten. Nowadays, electronic devices are capable of capturing human biometrics in a very convenient way. For example, fingerprint can be captured with a press on the sensor. Speech can be recorded by a microphone. Facial images can be shot by a CCD camera. As a result, people are willing to cooperate when biometric-based security systems are implemented. Besides, the "September 11th" incident has affected the public view on privacy and security. Before the incident, privacy was preferred. However, after the incident, the requirement for tighter security than before has desperately raised the needs for more exact identification and verification methods. This is why biometrics have gained wide acceptance nowadays.

In the field of security technologies, biometrics are defined as *measurable physical or behavioral characteristics of human beings*. In order to be applied to identify or verify human beings, the following criteria of a biometric have to be justified [24, 44, 78]:

- *Universality* means every person should have or can produce the biometric.
- *Uniqueness* means the difference between any two persons should be sufficiently distinguishable.
- *Permanence* means the biometric should not change drastically under environment or with time.
- *Collectability* means the biometric should be quantitatively measurable.
- *Acceptability* means people should be willing to use the biometric system.
- *Performance* specifies the achievable identification (verification) accuracy and resources needed to achieve acceptable accuracy.
- *Circumvention* means how easy it is to fraud the biometric system.

So far, many biometrics have been utilized for identification and verification. Physical characteristics include iris, fingerprint, hand-geometry, palm-print, hand veins, and

etc. [24]. Behavioral characteristics include signature, speech, gesture, and etc. [78]. In this thesis, due to the availability of capturing equipment, only fingerprint, hand-geometry and speech are used for performance evaluation.

1.2 General Concepts in Biometric Systems

1.2.1 Identification versus verification

A distinction between identification and verification should be made clear. An identification system, sometimes called a recognition system, answers the question "Who am I?", and a verification system answers the question "Am I the person I claim to be?" [24, 78].

In an identification process, a 'one-to-many' comparison is conducted via a search through the database of registered persons to identify or recognize a claimed person. Typical biometric identification process often consists of the following steps:

- Biometric data of the claimed person is captured.
- A search is conducted through the biometric database of registered persons to find out whether there are similar biometric data stored in the database.
- A decision upon whether the claimed person is a *registered* person (i.e. genuine user) or not (i.e. impostor) is made (like "Yes, he is Mr. X" or "No, he is not").

However, in a verification process, there is no need for such a search because the registered biometric data to be compared is provided when the person claimed the access. Only a 'one-to-one' comparison is conducted in this case. Typical biometric verification process follows the following steps:

- The person claims access by keying in a password or showing an ID card.
- Biometric data of the claimed person is captured.

- A comparison is carried out between the captured biometric data and the biometric data specified by the password or ID card. In this step, usually, a *matching score* that represents the similarity between two patterns is generated.

- A decision upon whether the claimed person is who he claims to be or not is made (like "Yes, he is" or "No, he is not") by making comparison between the matching score and a predefined *threshold*.

Identification problem is harder than verification problem because of the search process. However, one can easily convert the identification problem into multiple verification problems by making a comparison between the captured biometric data and all registered biometric data in the database. Hence, verification problem is the basic problem, and the focus of this research.

1.2.2 Performance measures of a verification system

The effectiveness of a verification system is always the first question: *is it possible that the system allows access to an unregistered person? How often does the system reject a truly registered person?* In this section, some performance measures of a verification system are discussed. A truly registered user is referred to as a *genuine user* and unregistered person as an *impostor* throughout the thesis.

Let s be the matching score and θ be a predefined threshold. Assume that the state of nature of the claimant is known (i.e. genuine user or impostor), and assume that if $s > \theta$, the final decision of the system is to accept the claimant. The criteria of a verification system are based on four probabilities:

- $FAR = P(s > \theta | impostor)$: False Acceptance Rate - the probability that the system accepts a user given that he is an impostor. In this case, an intruder is allowed to access the system. It is desirable that this probability is restricted to be less than a certain value (say, 10^{-5} means only one over one hundred thousand impostor trials may be accepted).

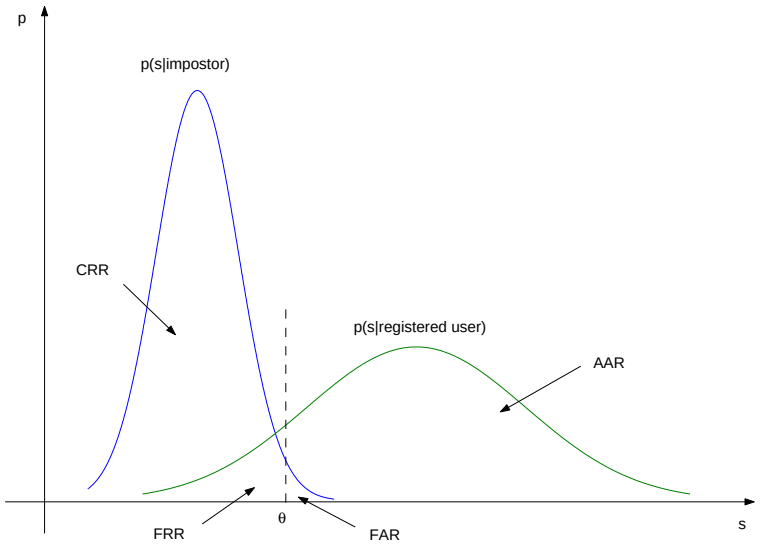


Figure 1.1: The hypothetic matching score distributions of genuine user and impostor, the arrows point to areas that represent four probabilities FAR, AAR, FRR and CRR.

- $AAR = P(s > \theta | genuine\ user)$: Authentic Acceptance Rate - the probability that the system accepts a user given that he is a genuine user. As the FAR is restricted to a certain level, the AAR is expected to be as large as possible, because AAR shows the friendliness of the system. Often, these two objectives contradict each other.

- $FRR = P(s < \theta | genuine\ user) = 1 - AAR$: False Rejection Rate - the probability that the system rejects a user given that he is a genuine user.

- $CRR = P(s < \theta | impostor) = 1 - FAR$: Correct Rejection Rate - the probability that the system rejects a user given that he is an impostor.

Although the four probabilities cannot be calculated exactly, they can be estimated experimentally when a large number of trials is conducted. Fig. 1.1 shows the areas that represent these four probabilities in a hypothetic case where the score distributions are normal with separated means. At each value of the threshold, the FAR and AAR specify a point in a two-dimensional graph. As the value of the threshold is changed, the FAR and AAR also change and the point moves along a curve which is

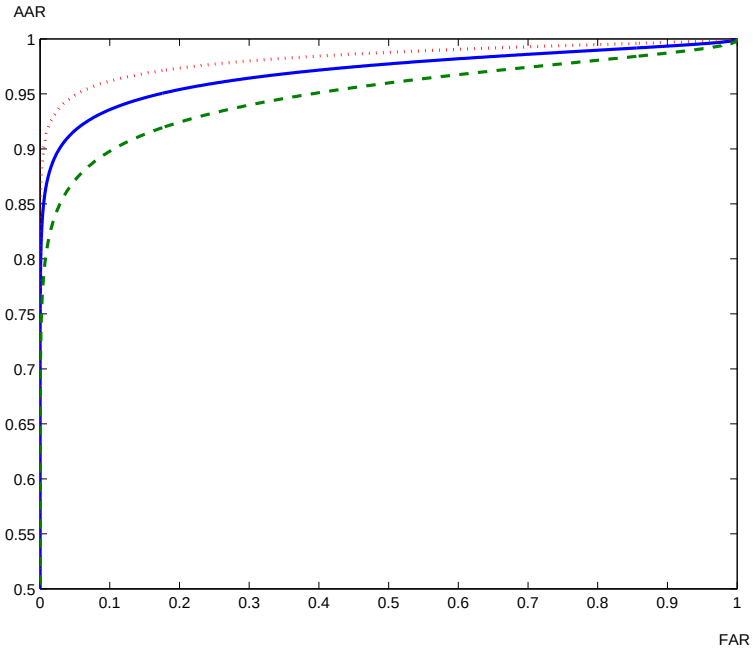


Figure 1.2: The ROC curves – thick line – corresponds to the above hypothetic case, dotted line – when two score distributions are moved farther apart, dashed line – when two score distributions are moved nearer with more overlapped region.

called the Receiver Operating Characteristics (ROC) of the verification system. Fig. 1.2 shows the ROC curves of the above hypothetic case and the cases when two score distributions (see Fig. 1.1) are moved farther apart (easier to classify) and nearer towards overlapping (more difficult to classify). As shown in Fig. 1.2, the thick line is below the dotted line and is above the dashed line. This means that the more accurately the system distinguishes genuine users and impostors, the higher the ROC curve is. Thus, ROC curve is an important measure showing the performance of a biometric verification system, and is used in this thesis.

1.3 Overview of Uni-Modal Biometric Verification Systems

Uni-modal biometric verification is a process involving measurement of a claimant's single biometric trait, and comparison with biometric templates of registered users. The outcome of this process is either an acceptance or a rejection depending on the degree of similarity between the claimant's biometric and the templates. The underlying steps of this process are shown in Fig. 1.3, and described as follows.

Biometric capture. First, biometric measurement of the claimant is measured using a specific biometric device. The biometric templates of the registered user can be achieved in the same way, except that they are usually measured much more carefully. Nowadays, fingerprints can be captured by electronic devices that are much more convenient than using black ink. Speech can be recorded using microphones. Faces and palm prints are sampled by video cameras. Often, these devices can be directly connected to a computer, which makes the data acquisition process more convenient than before [24, 78].

Feature extraction. Although raw data obtained as above can be fed into the database for future processing, usually a feature extraction process is performed and only some key features of the biometric are stored in the database to speed up the matching process. Feature extraction has two advantages. First, it reduces the space required to store biometrics of the registered users, i.e., it reduces the size of the database, and hence increases the speed to process the data. Second, careful selection of key features can, in fact, enhance the performance of the matching process [34]. Fingerprint features can be special points in the fingerprint image called minutia, which are, for example, endpoints, bifurcations, and etc. [22]. For speech, Linear Prediction Coefficients (LPC) is a powerful tool to extract the features [30]. For faces, Principal Component Analysis (PCA) has been used very effectively to reduce the storage size as well for as extracting useful features [10]. This technique is also called 'eigen-

face' technique. Certainly, these are only representative examples of feature extraction techniques. In chapter 2, more techniques are cited and discussed.

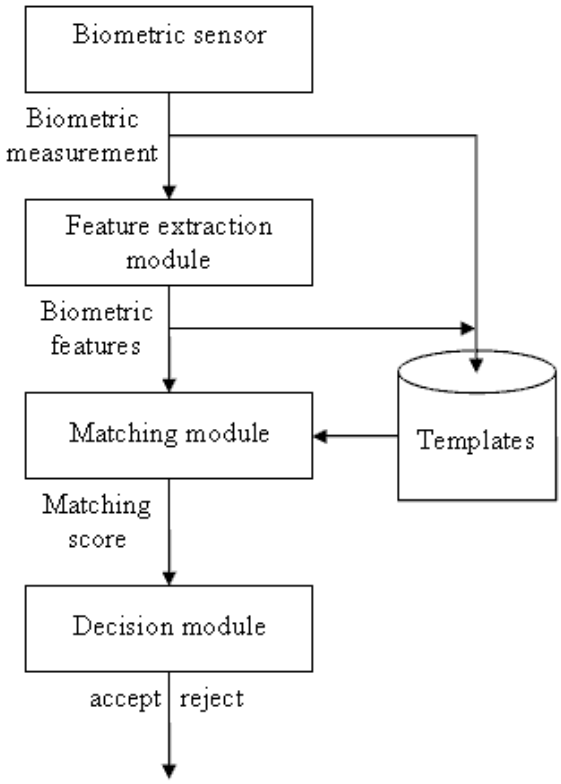


Figure 1.3: Uni-modal biometric verification

Feature matching. Once the key features in biometric measurement of the claimant are extracted, they are compared with those extracted from the registered users. Often, a similarity measure is defined between two sets of features. In the matching process, this similarity measure between biometrics of a claimant and a registered user is calculated. The outcome of this process is often a number which is the similarity measure itself or certain transformation of it. This outcome is also called the *matching score*.

Decision. At the final stage, the computed matching score is used in a decision module to give a final decision which is either an acceptance or a rejection (i.e. de-

cision that the claimant is a genuine user or not). In the simplest scheme, matching scores are compared with a certain threshold. If the matching score is greater (smaller) than the threshold, the final decision is an acceptance, otherwise, it is a rejection. The threshold is determined according to a certain error measure. For example, in high security systems, a threshold that results in small FAR is desirable (see Fig. 1.1). This threshold-based decision scheme is widely used in single [1] biometric verification systems [21]. However, simple comparison may not be the best scheme when multiple biometrics are used for verification purpose.

1.4 Overview of Multi-Modal Biometric Verification Systems

Multi-modal biometric verification is a process involving simultaneous measurement of several biometrics of the claimant to decide whether the claimant is a genuine user or an impostor. Multi-modal biometric verification is introduced due to limitations of uni-modal biometric systems [24, 44, 78]. First, individual biometric measurement may not be always in good condition. Fingerprints can be wet. Noise may interfere with speech recording. Sometimes, the users do not feel comfortable or even refuse to use certain biometric capturing device. For example, criminals are not usually willing to have their fingerprints or faces recorded. The handicapped may have lost their fingers or hands. Second, as performance of verification using different biometrics is different, there is hope that it is better to combine different biometrics to enhance the verification performance. In fact, multi-modal biometrics decision fusion for accurate identity verification has gained a lot more attention over recent years due to its performance improvement over uni-modal biometric verification (see e.g. [31, 41, 64]).

[1] We use interchangeably between the terms 'single biometric system' and 'uni-modal biometric system'.

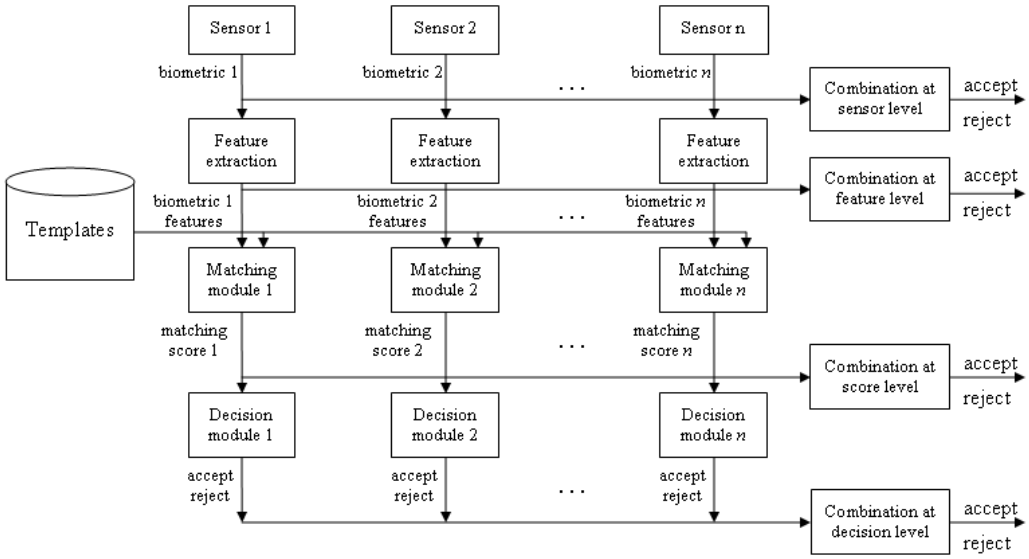


Figure 1.4: Multi-modal biometric verification

1.4.1 Approaches to multi-modal biometric verification

The process of combining multiple biometrics for verification is described in Fig. 1.4 [78]. The modules that process each biometric are as described in the previous section. The main difference between single and multiple biometric verification is the combination module. As shown in the figure, the combination module can be placed either before the matching phase or after it [24]. This results in different approaches to multiple biometric verification.

Before matching:

- **Sensor level combination.** The outputs of all biometric sensors are directly integrated for the decision process. No feature extraction results in very high dimensional input vector. Besides, information coming from different sensors is often incompatible. Thus, this approach is rarely used.

- **Feature level combination.** This approach treats all sets of features obtained from different biometrics as one single set of features. The problem of combining many biometrics at this level is similar to that above at sensor level but with

better compatible information.

After matching:

- **Score level combination.** The combination module takes in all matching scores generated by every biometric matching module as its inputs. These matching scores often form a real value input vector whose dimension is equal to the number of biometric modules.

- **Decision level combination.** The combination module takes in all decisions generated by every biometric decision module as its inputs. These decisions form a binary vector (‘1’ for genuine user, ‘0’ for impostor, vice versa) for combined decision.

This research focusses on score level combination since (i) combination in feature level does not utilize the matching modules which were developed for each biometric, (ii) the output at decision level is too simplified (i.e. ‘0’ or ‘1’) and crucial information may be lost. Combination in score level may overcome these problems [50].

1.4.2 Multi-modal biometric verification as a classification problem

In biometric authentication, a user when presented to the system is classified as either a *genuine user* or an *impostor*. Thus, the problem of combining the outputs of different biometric verification systems can be considered as a two-class classification problem. It has been observed that, even when each classifier (i.e. each uni-modal biometric verification system) is trained well, the misclassified patterns from different classifiers could be different [31]. This observation has fuelled hope of finding methods that can exploit the strength of each classifier. There are two different approaches to combine the outputs of classifiers: *classifier selection* and *classifier fusion* [37].

Classifier selection. In this approach, the outputs of different biometric modules (i.e. matching score) form a l-dimensional vector where l is the number of such modules. The l-dimensional space of such score vectors is, by some means, divided in to many regions. Each region is associated with a biometric module which is believed to perform better than other modules in that region. The decision is made in two steps: first, for each score vector the region and the biometric module associated with it are found; then a decision regarding the particular biometric module is taken to be the decision of the whole system in that particular operating region.

Classifier fusion. In this approach, the combination module takes the score vector as its input and produces a new matching score that is the basis for the decision of the whole system. From this point of view, the combination module can be trained from the observations of scores produced by each biometric module and the corresponding labels (i.e. genuine user or impostor). This learning task can be performed by applying any classifier that has been developed so far, ranging from the classical Bayesian classifier to decision trees, neural networks, support vector machines, and etc. Therefore classifier fusion is more flexible than classifier selection and this thesis concentrates on classifier fusion. Prior to multi-modal biometrics decision fusion, empirical comparison of several classifiers in terms of their classification accuracy, training time and storage requirement was conducted. A suitable classifier was then chosen for multi-modal biometrics decision fusion.

1.5 Motivation and Problem Statement

In [31–34, 41, 42, 64–66], it has been shown that combining multi-modal biometrics for verification purpose possesses higher accuracy than that of individual biometrics. However, there remain some problems when applying a parameterized classifier on multi-modal biometric verification system.

First, as new user registration can be a frequent process in a verification system,

it would be wise to develop an updating scheme that can easily adapt the system to new observations (i.e. data coming from new users) rather than retraining the entire system using old and new data whenever the enrolment process of a new user takes place. Therefore, an adaptive updating scheme for the applied classifier can enhance the model's performance in terms of time and memory storage when it is used in a multi-modal biometric verification system.

Second, from results reported in literature [29] and from the data collection process used in this research over a reasonably long period, some changes in biometrics data, especially the matching scores were noticed. These observations indicated that biometrics data should be considered as a sequence of data which varies over time. Scores drift over time can affect the performance of the verification system, especially if the system is trained only once and never gets updated from the data received from its day-to-day operation. Hence, an adaptive updating scheme would help the system adapt to changes, and therefore maintains or even enhances the verification performance.

Problem statement and scope

This thesis focusses on developing an adaptive updating scheme to track the performance of a multi-modal biometric verification system. As new observations may come from day-to-day operation of the system, the problem is to update the system's parameters so that it incorporates the new information into the system in an optimal manner. The updating formulation can be tuned so that the system can follow changes in the biometric data and maintains its verification performance.

1.6 Contributions of the Thesis

Main contributions of this work are listed as follows:

1. **Empirical evaluation of 9 classifiers [70] including RM model [63], its variants, KNN [77], SVM [45] and MLP [4] was conducted**.

- Comparison of 9 classifiers on 31 data sets obtained from UCI Machine Learning Repository [72] in terms of training time, storage requirement and classification accuracy was carried out.
- Unified selection of hyper-parameters in every classifier through 10-fold stratified cross validation was conducted. It was found that nominal data that have many discrete features are more difficult to classify than other data.
- A classifier that possesses good performance which is suitable for multi-modal biometrics decision fusion was selected.

2. **An adaptive updating scheme for multi-modal biometric verification was proposed.**

   - A recursive formulation to adapt the parameters of the system to newly registered patterns was proposed.
   - A stability limit of the algorithm was obtained.

3. **Empirical evaluation of the adaptive formulation using multi-modal biometrics data which varies over time was carried out.**

   - Collection of two fingerprint image data sets (one obtained over 20 weeks, the other over 30 weeks) was conducted.
   - Experiments on combination of fingerprint, speech and hand-geometry for user verification were conducted.
   - Evaluation of verification performance along with time, and with added noise was conducted.

1.7 Thesis Organization

The thesis is organized as follows. In chapter 2, a literature review on related works is presented. Firstly, different matching algorithms for fingerprint, speech and hand-geometry are discussed. Then, previous works on combination of multiple biometrics are briefly described in two categories: training based methods and non-training based methods. In chapter 3, extensive comparative experiments on several classifiers are reported. The experiments focus on performance of the classifiers in order to choose a suitable classifier for integrating different biometrics. In chapter 4, an adaptive updating scheme for a selected classifier is formulated for multi-modal biometric verification. Along with the formulation, other aspects such as implementation, stability of the algorithm are discussed as well. In chapter 5, experiments on two reasonably large biometric data sets which consist of fingerprint, speech and hand-geometry biometrics are reported. Discussion on the performance of the adaptive algorithm follows the experimental results. Finally, chapter 6 presents some concluding remarks and suggestions for future work.

Chapter 2

Literature Review

In this chapter, current research literature on biometric verification is discussed. First, representative works on uni-modal biometric verification related to this research's scope (fingerprint, speech and hand-geometry) will be covered. Second, previous works to combine different biometrics are discussed and divided categorically into non-training based methods and training based methods. Among the training-based methods, an important approach is the treatment of biometric combination problem as a two-class classification problem. From this point of view, many existing classifiers can be applied. Possible use of these classifiers for multi-modal biometrics application will be discussed in section 2.3.

2.1 Uni-modal Biometric Verification

2.1.1 Fingerprint verification

Among the various human biometrics, fingerprint is the most commonly used biometric for verification purposes. Due to the uniqueness of fingerprint, different identities can be distinguished with high accuracy (see e.g. S. Pankanti, et. al. [48]). Besides, fingerprints can be easily acquired via a simple finger press on the sensor. This has gained much user acceptability in adequate environments like offices.

Two fingerprint image samples can be matched manually by well-trained fingerprint experts, but this is a very slow process. To speed up the verification process, automatic fingerprint matching systems have been developed. Several approaches are reported in literature and can be divided into two categories: minutia-based matching and non-minutia based matching (see e.g. D. Maltoni et. al. [44], D. Zhang et. al. [78]).

- **Minutia-based matching:** In this approach, features like ridge endings and ridge bifurcations are extracted and their positions (coordinates in planar or polar system), their directions (the direction of the associated ridges) and the associated ridges are recorded. To compensate with deformations such as translation and rotation, A. K. Jain [22] performed an alignment step between two minutiae's ridges. Then, using elastic string matching algorithm, the corresponding minutia pairs were found, on which the matching score was based. Meanwhile, X. D. Jiang [27], by using the minutia in the neighborhood, computed the local features which consist of the position and direction of each minutia relatively to its k-nearest neighbors in order to obtain feature vectors which are invariant to translation and rotation. Also, global features consisting of the position in polar coordinates and the direction of each minutia with respect to the reference points were computed. The similarity of local features and global features between the input fingerprint and pre-stored templates formed the basis of the matching score. A. K. Hrechak [15] proposed that not only primitive features like ridge ending or bifurcation but also the compound features such as island, spur, crossover, bridge and short ridge be extracted. Other improvements in minutia-based matching algorithms used local alignment (see D. Lee, et. al. [40]), and orientation-improved minutiae (see L. Sha and X. Tang [59]).

  Although minutia-based matching is most commonly used, the disadvantage of this approach is that minutia (e.g ridge endings and bifurcations) are difficult to

be extracted reliably, especially from poor quality fingerprint images [29, 44]. In order to overcome these problems, robust methods which do not rely on minutia extraction have been implemented [44].

- **Non-minutia-based matching :** Optical correlation may be the earliest fingerprint matching approach (see e.g. F. Gamble, et. al. [12], K. Venkataramani and B. V. K. Vijaya Kumar [74]). This approach involves comparison of two fingerprint images pixel-wise or window-wise. Although many improvements have been introduced, comparison of images is still very time consuming. In [19], a framework, called *graph matching*, to convert a fingerprint image to a graph was proposed by D. K. Isenor and S. G. Zaky. The nodes of the graph represent ridges while the edges represent the joining points between ridges, and whether two ridges are neighbors of each other. Then a graph matching algorithm is performed in three steps: partitioning, refinement and scoring. In another work [21], A. K. Jain claimed that minutia-based methods faced problems such as different minutia's list length, and minutia's incapability to completely represent local ridge structures, and proposed that features can be extracted by applying Gabor filter to the input image in a sector-by-sector manner around a reference point defined as where the maximum curvature in concave ridges is obtained. This has provided equal length feature lists and simplified the matching step which involved only an Euclidean distance calculation.

2.1.2 Speech (Voice) verification

Speech verification is also easily accepted in normal working environment. The user is simply required to utter a word or a sentence to a micro-phone and the corresponding analog signal is sampled into digital version. If the sentence is fixed, it is called *text-dependent* speech verification. Otherwise, it is called *text-independent* speech verification.

Feature extraction in speech verification often involves computation of the Linear Predictor Coefficients (LPC). Other features like Reflection Coefficients (RC), Log-Area Ratios (LAR), and etc. can be computed from LPCs. Another popular feature which does not require LPC computation but utilizes Fourier transform is the Mel-Warp cepstrum [30]. This set of features can be reduced using Principle Component Analysis (PCA). As a result, a sequence of feature vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M)$ are extracted from the speech sample through window (frame) sampling. Finally, the matching score is computed through comparison between two sequences of feature vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M)$ and $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$.

While the dissimilarity $d(\boldsymbol{x}_{i(k)}, \boldsymbol{y}_{j(k)})$ between two feature vectors can be simply computed using Euclidean distance, Mahalanobis distance or Bhattacharyya distance [30], computing dissimilarity between two sequences of feature vectors requires mapping between two sequences, and is hard to implement. Some representative approaches to compute the matching score between two sequences of feature vectors reported in literature are: Dynamic Time Warping algorithm (DTW), Vector Quantization source modeling (VQ), Nearest Neighbors method (NN) and Hidden Markov Models (HMM).

- **DTW algorithm [56]:** A so-called warp function $F = (c(1), \ldots, c(K))$ where $c(k) = (i(k), j(k))$ (i.e. the mapping function maps $\boldsymbol{x}_{i(k)}$ onto $\boldsymbol{y}_{j(k)}$) is computed through dynamic programming technique in order that the error function $E = \sum_{k=1}^{K} d(\boldsymbol{x}_{i(k)}, \boldsymbol{y}_{j(k)})$ achieves its possible minimum. This warping error function is the basis of the matching score.

- **VQ source modeling [61]:** From each registered user's training data, a VQ codebook $\boldsymbol{C}$ is generated through standard clustering technique such as k-mean clustering. The codebook $\boldsymbol{C}$ contains the centroids of these generated clusters. The matching score is computed based on distance between the input vector and the nearest code word in $\boldsymbol{C}$ as follows $E = \sum_{i=1}^{M} \min_{\boldsymbol{y} \in \boldsymbol{C}} d(\boldsymbol{x}_i, \boldsymbol{y})$, where

$(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M)$ is the input sequence of feature vectors, $\boldsymbol{y}$ is the nearest code word in $\boldsymbol{C}$ with respect to $\boldsymbol{x}_i$.

- **Nearest Neighbors method [14]:** This method is an attempt to combine DTW matching and VQ modeling. It stores all the registered users' training data and computes the nearest distances between the claimant's sequence and all sequences stored in the database. The distances are then averaged to form the matching scores. This method is one of the most memory and computation intensive methods.

- **HMM method [51]:** Generally, HMM models each registered user by a number of states and the probability to move from one state to another. Given the models (computed from the training data), the probability that the claimant's speech is generated by each model is computed and used for obtaining the matching score. Details of application of HMM on speech recognition can be found in [51].

Recently, speech verification based on Gaussian Mixture Models (GMM) has been proposed. S. Z. Li used AdaBoost to enhance the GMM approach (see S. Z. Li et. al. [62] for more details). In a survey [30] on speech recognition, HMM-based methods are reported to be comparable to VQ methods in text-independent testing and are recognized to be superior to other methods in text-dependent testing.

2.1.3 Hand-geometry verification

Among several factors that raised the applicability of a certain biometric, user acceptability seems to be the most important ones. Hand-geometry, although its verification performance is average, is generally more acceptable to users as the image collection and sensing process are very simple. Besides, in some situations, it is an advantage that hand-geometry is not very distinctive because a very distinctive biometric like fingerprint may raise the problem of revealing users' privacy i.e. linked to criminal and

identity records. In such cases, hand-geometry is a good choice. There have been relatively few reports on hand-geometry verification even though it is among the earliest automated biometrics. Followings are some of the most recent approaches:

- **Prototype hand-geometry based:** In [26], an image of size 640×480 that consists of top-view and side-view of the hand is used. The intersections between sixteen predefined lines and the edges of the hand images is calculated as the extracted features. A matching score between two hand images is calculated based on Euclidean or weighted Euclidean distances.

- **Deformable matching:** In [25], before the matching score is calculated, two hand edge contours are aligned. By running an exhaustive search for correspondence points between two images, a transformation matrix can be computed. Using this transformation matrix to match every point in a contour with those on the other image results in a matching score (in the paper, it is called mean alignment error).

- **Hand-geometry measurement:** In [53], similar to [26], images are taken from the top-view and side view of the hand. However, a different set of features consisting of the width of each finger at various positions, the height of the palm, etc. is computed. For the matching process, either Euclidean distance or Hamming distance can be applied. Meanwhile for identification, each user is modelled using Gaussian Mixture Model or a Radial Basis Function network. The experiments showed that Gaussian Mixture Model achieves highest accuracy but it requires high computational cost and storage for the templates.

Although the verification accuracy of hand-geometry is not very high, it is expected that by including it in a multi-modal biometric verification system, good performance can be achieved.

2.2 Multi-modal Biometric Verification

2.2.1 Different implementations in multi-modal biometric verification

A multi-modal biometric verification systems can be implemented in various ways. The purpose of such implementation can be either for high security or for better convenience. These two objectives often contradict each other. If all biometrics must be verified concurrently, high security can be reached albeit at the expense of user convenience. According to S. Prabhakar, et. al. [50], various available implementation schemes in multi-modal biometric verification can be classified as follows:

- **Multiple sensors system:** This system consists of different capturing devices for the same biometric, such as optical sensors, ultrasound sensors, and solid-state sensors to capture fingerprint images [44].

- **Multiple matchers system:** There are many matching algorithms for a certain biometric (see previous section). Each algorithm can generate a matching score (i.e. similarity measure) and confidence level. This system implements several matching algorithms for a biometric (for example, fingerprint), and combines the outputs of these algorithms following certain rules to achieve a final decision.

- **Multiple units system:** Multiple biometric parts of the same biometric type are captured (for example, index and middle fingers, or left and right iris) and matched simultaneously.

- **Multiple impressions system:** This system allows several enrollments and several inputs for verification. The purpose is to extract the most reliable features from the user's biometric. As a result, the verification is more reliable.

- **Multiple biometrics system:** In this system, different biometrics are captured (for example, fingerprint, speech and hand-geometry) and matched using dif-

ferent matching algorithms. The difference between this system and multiple matchers system is that multiple biometrics system uses many biometrics simultaneously rather than a single biometric. This thesis adopts this multiple biometrics system.

In order to build a multiple biometrics system, the combining module has to implement a combination method. The combination methods can be implemented at various levels: sensor level, feature extraction level, decision level, and matching score level (see e.g. [23]).

- **Sensor level**: The raw data obtained from different sensors can be combined to make the final decision. For example, face images recorded by different cameras have been combined to form a single face image [20]. However, combination at sensor level require that the raw data obtained from the sensors must be compatible. Since this may not always be possible, in this research, combination at sensor level was not considered.

- **Feature extraction level**: It is difficult to combine features extracted from different biometrics and different feature extraction algorithms as they are often either inaccessible or incompatible. Especially, in commercial biometric systems, access to the features extracted by the built-in algorithm is often not allowed [23].

- **Matching score level**: Combination at matching score level can overcome problems of combination at other levels. The matching scores are accessible as outputs of different biometric matching algorithms are often the degree of certainty that the biometric patterns belong to 'genuine' class or 'impostor' class [31–34, 41, 42]. Additionally, the matching scores can be normalized to avoid the incompatibility between them [23].

- **Decision level**: Combination at decision level is too rigid because the output at

decision level is too simplified (i.e. '0' or '1') and crucial information may be lost. Major combination approaches at this level are majority voting (L. Lam and C. Y. Suen [38]), AND and OR rule (J. Daugman [9]), and Behavior Knowledge Space (Y. S. Huang and C. Y. Suen [18]).

Generally, combining methods can be divided into two types: non-training based methods and training-based methods. In non-training based methods, it is often assumed that the outputs of individual classifiers are the probabilities that the input pattern belongs to a certain class. The training based methods often do not require this assumption and can operate directly on the matching scores generated by biometric verification modules.

2.2.2 Non-training based methods

An intuitive approach to combine multiple biometrics is by the use of simple combination rule based on the matching scores generated by different biometric classifiers to make the final decision. J. Kittler, et. al. [31] states that the *product rule* is originated from the optimal Bayes classification rule under the assumptions that (i) the matching scores are estimates of the a-posteriori probabilities that the provided claimant belongs to each class (i.e. 'genuine user' or 'impostor'), and (ii) there is independence between the classifiers. This provided the theoretical basis for other combination rules like *sum*, *min* and *max* rules. Under these assumptions, the simple combination rules can be formulated as follows

Product rule: find class ω_k that maximizes $P(\omega_k)^{1-L} \prod_{i=1}^{L} P(\omega_k|x_i)$.

Sum rule: find class ω_k that maximizes $(1-L)P(\omega_k) + \sum_{i=1}^{L} P(\omega_k|x_i)$.

Max rule: find class ω_k that maximizes $(1-L)P(\omega_k) + L \max_{i=1}^{L} P(\omega_k|x_i)$.

Min rule: find class ω_k that maximizes $P(\omega_k)^{1-L} \min_{i=1}^{L} P(\omega_k|x_i)$,

where L is the number of classifiers, $\omega_1 =$'genuine user', $\omega_2 =$'impostor', $P(\omega_k|x_i)$ is the output of the i-th classifier (i.e. the estimated a-posteriori probabilities that the

provided claimant belongs to each class).

Supporting this approach, L. Hong, et. al. [42] proved that under the independency of individual classifiers assumption, there is always a combination rule which results in smaller error rate than that of individual classifiers.

Other non-training based approaches are: majority voting (L. Lam and C. Y. Suen [38]), AND and OR rule (J. Daugman [9]), highest rank, Borda count (T. K. Ho, et. al [16]). L. I. Kuncheva, et. al. [37] has studied the limits of majority voting rule. She showed that majority voting can improve the performance even when the classifiers are dependent.

2.2.3 Training based methods

Under this approach, the combination module exploits the training data to learn the behavior of each biometric classifier, and therefore, can achieve better performance than non-training based methods when the data are representative.

S. Prabhakar and A. K. Jain [50] argued that the independence assumption may not be true when different matching algorithms of the same biometric trait are combined. This suggests a combination method that can learn the behavior of the biometric classifiers from training data. S. Prabhakar and A. K. Jain proposed a scheme based on non-parametric density estimation of the scores. They showed that the method is optimal in the Neyman-Person decision sense.

J. Kittler and K. Messer [32] applied two trainable classifier fusion methods, namely the Decision Templates of L. I. Kuncheva, et. al. [36] and the Behavior Knowledge Space of Y. S. Huang and C. Y. Suen [18], to combine face and speech data for verification purpose.

Decision Templates [36] tries to distinguish the classifiers' responses to 'genuine user' and 'impostor' under the assumption that the support scores of 'genuine user' and 'impostor' classes will form two clusters with separated means. The support scores

of individual classifiers can be the a-posteriori probabilities (i.e. $P(genuine|x)$ and $P(impostor|x)$) or more directly, the matching scores. The support scores then form a decision profile matrix as follows:

$$D(x) = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \\ \cdots & \cdots \\ s_{L1} & s_{L2} \end{bmatrix}, \tag{2.1}$$

where L is the number of biometric classifier, s_{i1} and s_{i2} are the support scores for 'genuine user' and 'impostor', respectively. By averaging the decision profile matrices of 'genuine user' and 'impostor' samples, the decision templates for each class can be calculated. Any claimant's decision profile is compared to these decision templates using some similarity distance (in [32], Euclidean distance was used) to generate the soft class labels.

Behavior Knowledge Space method, like its name, builds a space that indicates the behavior of individual biometric classifiers according to the training data. In this method, each classifier generates the exact class of the claiming user (i.e. '1' for 'genuine user' and '0' for 'impostor'). L classifiers therefore generate a binary vector $\boldsymbol{x} \in \{0,1\}^L$. Each binary vector will index a bin in the discrete space $\{0,1\}^L$. Each bin is associated to the class that has more samples from the training set falling in it. The space with this association is called Behavior Knowledge Space. Any claiming user's binary vector will be classified as the class that is associated to the bin in which it falls.

In the survey [32], J. Kittler and K. Messer compared Decision Templates, Behavior Knowledge Space and other combination rules like product, sum, min, max rules in an application that combines face and speech verification. The survey showed that these fusion methods give better performance than each single biometric classifier. However, the experiments also showed that no significant evidence has been found

that trainable methods like Decision Templates and Behavior Knowledge Space can outperform simple combination rules (i.e. product, sum, min, max) in all test cases. A. Ross and A. K. Jain [54] observed that sum rule's performance is better than that of decision trees and linear discriminant function while combining face, fingerprint and hand-geometry. This suggests that it is nessesary to search for more powerful trainable combination methods for possible nonlinear decision hyper-surfaces.

As mentioned in chapter 1, multiple biometric verification can be considered as a two-class classification task. From this point of view, many classifiers proposed in literature can be applied to build the combination module. In next section, several methods which follow this point of view will be discussed.

2.3 Multi-modal Biometric Verification as a Classification Task: Related Works

An important aspect of this approach is the normalization of matching scores from different biometric classifiers. As seen in chapter 1, once the user claimed his identity, a score vector $\boldsymbol{x}$ of size $L \times 1$, where L is the number of classifiers, is generated and presented to the combination module as a feature vector. The elements of this vector may have different ranges, means and deviations due to different generating mechanisms of these classifiers. To make sensible decision and benefit the combination performance, these scores should be normalized before being presented to the combination module. R. Brunelli and D. Falavigna [5] normalized the score to the range $(0, 1)$ by means of hyperbolic tangent function. In [43], C. L. Liu et. al. even classified different output functions of classifier (i.e. linear, log-likelihood, exponential, sigmoid) and gave each output function type a special way to transform the scores (or 'confidence' in the paper).

In [3], J. Bigun, under an assumption of normal distribution on the bias of each

classifier opinion and via Bayes theory, estimated the means $M_i, i = 1 \ldots L$ and the variances $V_i, i = 1 \ldots L$ of the biases of the classifiers' matching scores with respect to the true matching scores ('Yes' or 'No') using the training data. After the classifiers' matching scores have been normalized by adding the means and variances of the bias, these normalized matching scores form the basis of the combination matching score following the equation:

$$M = \frac{\sum_{i=1}^{L} \frac{M_i'}{V_i'}}{\sum_{i=1}^{L} \frac{1}{V_i'}}, \tag{2.2}$$

where M_i' and V_i' are the normalized matching score and variance. Equation (2.2) shows that the final matching score will be calculated as the weighted average of all classifiers' matching scores. This computation prefers classifier's matching scores to be with high accuracy i.e. classifiers having small variance V_i'. In a recent study [49] on the relation between the Equal Error Rate (EER), and the correlation cum variance of classifiers, N. Poh Hoon Thian and S. Bengio showed that the EER can be modeled as a function of correlation, variance and difference between genuine and impostor means. As a result, in order to achieve a low EER, small correlation, small variance, and a large mean difference are needed.

In [7], V. Chatzis, et. al. used many classical classification techniques to combine the matching scores of five biometric modules: four are on face verification and one on speech verification. The studied techniques include k-means clustering (KM), fuzzy k-means clustering (FKM), fuzzy vector quantization (FVQ) and median radial basis network (MRBF). The experiments showed that MRBF achieved best performance (i.e. lowest FAR and FRR) when combining 2 modalities consisting of face and speech. The structure of MRBF network suggests that network-type classifier with sigmoidal kernel function can be used for multi-modal biometrics fusion. In chapter 3, many other types of kernel function that can be used for classification have been investigated.

S. B. Yacoub, et. al. [76] applied Support Vector Machines (SVM), minimum cost Bayesian classifier, Fisher's linear discriminant, C4.5 decision trees and Multi-Layer

Perceptron (MLP) to combine face and speech biometrics for verification. In their experiments, SVM with polynomial kernel and Bayesian classifier gave the best results. The Bayesian classifier requires data modeling (i.e assumption on the parametric distribution) while SVM does not require so. Besides, for low FA rates (say, less than 1%), MLP has the lowest FR rates. This again suggests that the network-type classifier can be applied with good performance. A disadvantage of MLP is its iterative training process, especially when the network has many parameters and when high test accuracy is required [63].

An approach to bypass the iterative training process is to use linear formulation, because a system of linear equations can be solved effectively. In [63], a reduced multivariate polynomial model (RM) was introduced. The model consists of a reduced number of polynomial terms and a single-step regularized solution. In [64], the RM model has been used to combine three biometrics namely, fingerprint, speech and hand-geometry for verification and many common classifiers. The combination outperformed individual biometrics. However, one disadvantage of RM model is that when new training samples arrive (new user registration), it requires retraining the model using the entire data set. This suggests that if an adaptive formulation can be derived, RM model can be a very effective tool in the field of multiple biometrics fusion.

None of the discussed combination methods have considered the situation where the system needs to be updated according to biometric data obtained from day-to-day operations or where any change in the matching scores can affect the verification performance. In uni-modal biometrics, regarding a study on the usage of online fingerprint verification system, X.D. Jiang [29] suggested that the extracted fingerprint features should be considered as a sequence of data which varies over time. As a result, he proposed an adaptive method that can update the system from fingerprint samples obtained from day-to-day usage in order to enhance the verification performance. Hence, it is expected that, in a multi-modal biometric verification system, an

adaptive updating scheme can maintain or enhance the verification performance of the system over time.

2.4 Summary

In this chapter, many biometric verification techniques including fingerprint, speech and hand-geometry are discussed. For multi-modal biometric verification, the idea of considering the problem as a two-class classification problem has opened the door for many classifiers to be applied in this field. To choose a suitable classifier, it is necessary to evaluate and justify the classifiers in terms of their training time, memory storage and classification accuracy. Training time refers to the speed at which the registration process can be performed. Memory storage means how large the space for the combination module is. Most importantly, classification accuracy specifies how reliable the multi-modal biometric verification system is. In next chapter, several classifiers are evaluated on real-life data sets. The experiments will be reported along with a discussion to choose a suitable classifier as a basis for the multi-modal biometrics decision fusion algorithm developed in this thesis.

Chapter 3

Evaluation of Classification Tools

Considering the problem of multi-modal biometric verification as a classification problem, an intuitive question arises: which classifier should be used among many classifiers available? To answer this question, an evaluation of classifiers in two aspects which are accuracy and efficiency is needed. Accuracy means how well each classifier distinguishes members of 'genuine' and 'impostor' classes. Efficiency means how costly in term of time and storage each classifier needs to perform its task. Besides, as this research's purpose is to find an efficient adaptive updating scheme for multi-modal biometric verification, the classifier of choice should be able to be formulated in adaptive form. Those aggregated classifiers like boosting and bagging [2] are thus not considered in this research.

In this chapter, three important classifiers: k-nearest neighbor, neural networks and support vector machines are discussed along with a recently developed polynomial classifier, the RM model [63]. Beside the original RM model, other extensions using hyperbolic basis functions like $tanh(x), sinh(x), cosh(x)$ are also introduced and discussed.

Based on the theory of optimal classifier (i.e. Bayes classifier), the statistical approach in pattern classification has received significant attention (see e.g. [10]). Generally, under this approach, the statistical properties (e.g. probability distributions) are

first recovered from the training set. Then a decision rule is derived using Bayes law in order to minimize the overall risk. A classical algorithm applying this approach is the k-nearest neighbors algorithm (kNN). This algorithm, which will be included in the experiments, is widely used due to its simplicity and its convergence to an optimal classifier as k increases [10].

Different from the neural network models, the Support Vector Machines (SVM) apply the structural risk minimization principle [73]. By maximizing the margin between pattern classes, SVM minimizes the bound on the generalization error. The SVM training process is performed through solving a quadratic programming problem which is an iterative process. SVM has shown its very good performance on many applications. For example, in digits recognition, SVM with polynomial kernel is reported to achieve highest accuracy rate on NIST database [73].

Recently, a reduced multivariate polynomial model (RM) has been proposed by K. A. Toh [63, 66] for classification. The model reduces the number of parameters in the full multivariate polynomial while still preserves the classification capability. It is reported that RM model possesses better average accuracy on 42 UCI data sets than other classifiers including SVM. The RM model has also been applied in multi-modal biometric verification [64, 65].

In next section, a brief review of the classifiers mentioned above is given. Then in section 3.2, extensive experiments on these classifiers are carried out to compare classification performance of these classifiers [70]. The experiments used 31 data sets (different from the 42 data sets in [63]) taken from the UCI database [72].

3.1 Commonly Used Classification Tools

3.1.1 Support Vector Machines (SVM)

While conventional classifiers follow the empirical risk minimization principle (i.e. an error measure based on the training set), Support Vector Machines [73] follow the principle of structural risk minimization. This principle states that good generalization ability can be achieved by minimizing the bound on the generalization error.

Let the data set D, which consists of N data points, be divided into two classes labelled as $+1$ and -1 respectively. The function that maps the data points to their class labels can be expressed as:

$$f_{SVM}(\boldsymbol{\alpha}, \boldsymbol{x}) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right), \tag{3.1}$$

where $K(\boldsymbol{x}, \boldsymbol{y})$ is a positive definite symmetric function called the kernel function. $sgn(\cdot)$ is the sign function and b is a bias estimated from the training set. The parameter $\boldsymbol{\alpha}$ is the solution of the following quadratic programming (QP) problem:

$$\begin{cases} max_{\boldsymbol{\alpha}} J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \\ \text{with the constraints:} \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } 0 \leqslant \alpha_i \leqslant C, i = 1, 2, \ldots, N. \end{cases} . \tag{3.2}$$

In non separable case, the constant C must be set to a given value. Choosing a value for C can be done through an empirical search like cross-validation.

The kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ defines the nature of the decision surface that separates the data. This function should satisfy some constraints in order to be applicable (Mercer's conditions [57]). A typical and commonly used kernel is the polynomial kernel $K(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^T \boldsymbol{y} + 1)^d$, where d is a positive integer defining the order of the polynomial, γ is a real number that normalize the inputs. In applying SVM with polynomial kernel, the parameters d, γ, C (so called 'hyper parameters') should be

defined (e.g. via cross-validation) before classifying unknown data.

The modification of SVM for multi-class case is done through integrating many single-output SVM classifiers in one-versus-the-rest scheme [57] or pairwise scheme [35]. In our experiments, a SVM Matlab toolbox [45] which has already implemented SVM with polynomial kernel for multi-class case is used. SVM with the polynomial kernel was chosen in the experiments because (i) in [73] the polynomial kernel showed good performance as compared to other kernels and (ii) RM and its extensions also applied multivariate polynomials in their implementation.

3.1.2 k-Nearest Neighbor classifier (kNN)

The k-nearest neighbor decision rule states that an incoming pattern $\boldsymbol{x}$ is assigned to the class which most frequently appeared among the k nearest samples (i.e. data points in the training set) [10]. This decision rule is based on the k-nearest neighbor estimation of the a posteriori probability. As k goes to infinity, the kNN estimation approaches the true probability [10]. Therefore, the kNN error rate approaches the Bayes error rate which is optimal. Analysis of kNN has shown that even when $k = 1$, the error rate is less than twice the optimal (Bayes) error rate [10]. Because of this behavior and the simplicity of the method (i.e. voting among nearest neighbors), kNN decision rule is widely used.

Let $D = \{(\boldsymbol{x}_i, y_i) | \, y_i \in \{1, \ldots, N_c\}, i = 1, \ldots, N\}$ be the training set of a classification problem with N_c classes and let $\boldsymbol{x}$ be a pattern to be classified. The kNN decision rule requires selection, from the set D, of k samples which are nearest to $\boldsymbol{x}$ in the sense of distance. These nearest samples form a collection $S = (y_{n_i} | i = 1, \ldots, k)$ of class labels that appear nearest to the pattern $\boldsymbol{x}$. The final decision for the class label of $\boldsymbol{x}$ is to choose among S the class label that appears most frequently. The measurement of distance can be carried out by any distance function $d(\boldsymbol{x}, \boldsymbol{y})$ which satisfies conditions of a metric distance (i.e. non-negativity, reflexivity, symmetry and trian-

gle inequality [10]). However, for simplicity, one often uses the Euclidean distance even though this distance can be strongly affected when the inputs are scaled. The Euclidean distance is defined as $d(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{x} - \boldsymbol{y}||_2$ where $||\cdot||_2$ is the L_2 norm. The kNN decision rule was included in the experiments using a Matlab implementation from [77].

3.1.3 Multi-Layer Perceptron (MLP)

Multi-layer perceptron (MLP) is an important class of neural networks [4]. Generally, a MLP consists of basic computational elements (nodes) which are named as input layer, hidden layers and output layer. As the activation functions in each computational nodes are smooth (e.g. a logistic function), MLP can be trained by the error back-propagation algorithm [55], a very well known training algorithm. According to the universal approximation theorem (see e.g. [13, 17, 58]), MLP possesses the capability to perform nonlinear input-output mapping up to any degree of accuracy (uniform approximation) provided that it has enough hidden nodes. However, this is only an existence theorem meaning that so far, there is no explicit way to determine this number of hidden nodes. With this background of approximation capability, MLP is widely used for pattern classification. In our experiments, a MLP with one hidden layer from Matlab's Neural Network toolbox [46] was used.

3.1.4 Reduced Multivariate polynomials (RM)

Grounded on Weierstrass's approximation theorem, the Multivariate Polynomial (MP) possesses the universal approximation capability (see e.g. [17, 58]). A general form of MP is a linear combination of all possible polynomial (product) terms $x_1^{n_1} x_2^{n_2} \cdots x_l^{n_l}$ with $n_1, n_2, \ldots n_l$ vary such that $\sum_{j=1}^{l} n_j \leqslant r$. The number r is the degree of the polynomial and K is the total number of weighting parameters α_j (i.e. the number of product terms). However, a full multivariate polynomial faces the problem of parame-

ter explosion. The number of parameter, K [58] is given by

$$K = \begin{pmatrix} l+r \\ r \end{pmatrix}, \tag{3.3}$$

which grows exponentially as l and r increase. In [63, 66], a reduced multivariate polynomial model, which has much less parameters while keeping crucial polynomial terms, was proposed as follows:

$$\begin{aligned} f_{RM}(\boldsymbol{\alpha}, \boldsymbol{x}) &= \alpha_0 + \sum_{k=1}^{r}\sum_{j=1}^{l} \alpha_{kj} x_j^k + \\ &+ \sum_{k=1}^{r} \alpha_{rl+k} (\sum_{j=1}^{l} x_j)^k + \sum_{k=2}^{r} (\sum_{i=1}^{l} \alpha_{r(l+1)+(k-2)l+i} x_i)(\sum_{j=1}^{l} x_j)^{k-1}. \end{aligned} \tag{3.4}$$

The number of polynomial terms in (3.4) now grows linearly over the degree and the number of inputs under the relationship $K = 1 + r + l(2r-1)$. To stabilize the solution for least squares error, a regularization can be performed [63]. The criterion function to be minimized is thus:

$$\boldsymbol{J} = ||\boldsymbol{y} - \boldsymbol{F}^T\boldsymbol{\alpha}||_2^2 + b||\boldsymbol{\alpha}||_2^2, \tag{3.5}$$

where $\boldsymbol{y} = [y_1, y_2, \cdots, y_N]^T$ is the target output vector and $\boldsymbol{F} = [\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, \cdots, \boldsymbol{f}_N^T]^T$ with $\boldsymbol{f}_i$ being the row vector of all polynomial terms in (3.4) which is applied to the i-th ($i = 1, ..., N$) training sample. The estimated output is $\hat{\boldsymbol{y}} = \boldsymbol{F}\boldsymbol{\alpha}$ and the solution for $\boldsymbol{\alpha}$ that minimize $\boldsymbol{J}$ is

$$\boldsymbol{\alpha} = (\boldsymbol{F}^T\boldsymbol{F} + b\boldsymbol{I})^{-1}\boldsymbol{F}^T\boldsymbol{y}, \tag{3.6}$$

where b is a regularization parameter (b is usually chosen to be a small value, say 10^{-4} [63], for stability and not introducing much bias).

As an example, consider the case where $r = 2$ and $l = 2$, we have

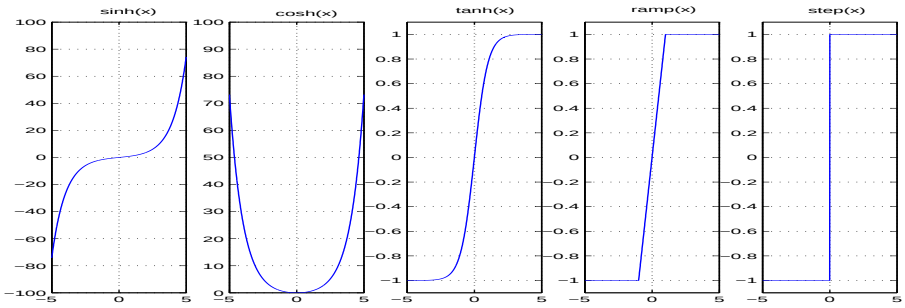


Figure 3.1: Basis functions: $sinh(x)$, $cosh(x) - 1$, $tanh(x)$, $ramp(x)$ and $step(x)$

$$\boldsymbol{F} = \begin{bmatrix} 1 & x_{1,1} & x_{2,1} & x_{1,1}^2 & x_{2,1}^2 & (x_{1,1}+x_{2,1}) & (x_{1,1}+x_{2,1})^2 & x_{1,1}(x_{1,1}+x_{2,1}) & x_{2,1}(x_{1,1}+x_{2,1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,N} & x_{2,N} & x_{1,N}^2 & x_{2,N}^2 & (x_{1,N}+x_{2,N}) & (x_{1,N}+x_{2,N})^2 & x_{1,N}(x_{1,N}+x_{2,N}) & x_{2,N}(x_{1,N}+x_{2,N}) \end{bmatrix}, \tag{3.7}$$

where the first and the second index of $x_{j,k}(j = 1, 2, k = 1, \ldots, N)$ specify the number of inputs and the number of samples, respectively. The output vector, $\boldsymbol{y}$, is known as the class label vector. For 2-class problems, the output, $y_k(k = 1, \ldots, N)$, is set to be '0' for a class and '1' for the other. Extension of this reduced model to multi-class case can be done through winner-take-all scheme [63] which is described shortly. Let N_c be the number of classes, then each training sample that belongs to class i is associated to a target vector of size $1 \times N_c$ consisting of all zeros but an '1' at the i-th element. Stacking the target vectors will form a target matrix $\boldsymbol{Y}$. The solution for the weighting parameters is

$$\boldsymbol{\Lambda} = (\boldsymbol{F}^T\boldsymbol{F} + b\boldsymbol{I})^{-1}\boldsymbol{F}^T\boldsymbol{Y}, \tag{3.8}$$

which is still similar to equation (3.6) except that the parameters $\boldsymbol{\Lambda}$ is a matrix of size $K \times N_c$ rather than a vector. The estimated output is $\hat{\boldsymbol{Y}} = \boldsymbol{F}\boldsymbol{\Lambda}$ where the maximum values in each row indicate the class label of each pattern.

3.1.5 Hyperbolic function networks (SINH, COSH and TANH)

The sigmoidal, hyperbolic and Gaussian functions have been widely used in neural network structures as nonlinear discriminant or activation functions. It has been shown that linear combination of perceptron basis functions is capable of approximating any function of interest to any desired accuracy provided that sufficiently many hidden variables are available (see e.g. [17]). The good approximation and classification capabilities of these networks are usually impaired by the tedious iterative training procedure due to the nonlinear formulation of learning parameters. Moreover, the iterative search does not guarantee convergence to desired optimal solution. In [65], linear combination of hyperbolic function network was shown to be useful for multi-modal biometrics decision fusion. It is shown briefly below how linear combination of hyperbolic functions can approximate those nonlinear parameters within the perceptron basis function for pattern classification.

Equations (3.9) - (3.14) show observations on some basic properties of product and power terms of the following hyperbolic functions: $sinh(x)$, $cosh(x)$ and $tanh(x)$. From (3.9)-(3.11), it can be seen that functions with small signal width or period can be expressed in terms of the sum of product and power terms of those with large signal width. Since $sinh(x)$ and $cosh(x)$ are related by $cosh^2(x) - sinh^2(x) = 1$, these hyperbolic functions can all be expressed in terms of their own original larger signal width functions. For cases with non-integer multiples of signal widths, (3.12)-(3.14)

can be applied for further dilation or contraction of signals.

$$\left.\begin{aligned} sinh(2x) &= 2sinh(x)cosh(x) \\ sinh(3x) &= 3sinh(x) + 4sinh^3(x) \\ sinh(4x) &= 8sinh^3(x)cosh(x) + 4sinh(x)cosh(x) \end{aligned}\right\} \tag{3.9}$$

$$\left.\begin{aligned} cosh(2x) &= 2cosh^2(x) - 1 \\ cosh(3x) &= 3cosh^3(x) - 3cosh(x) \\ cosh(4x) &= 8cosh^4(x) - 8cosh^2(x) + 1 \end{aligned}\right\} \tag{3.10}$$

$$\left.\begin{aligned} tanh(2x) &= \tfrac{2tanh(x)}{1+tanh^2(x)} \\ tanh(3x) &= \tfrac{3tanh(x)+tanh^3(x)}{1+3tanh^2(x)} \\ tanh(4x) &= \tfrac{4tanh(x)+4tanh^3(x)}{1+6tanh^2(x)+tanh^4(x)} \end{aligned}\right\} \tag{3.11}$$

$$sinh(x \pm y) = sinh(x)cosh(y) \pm cosh(x)sinh(y), \tag{3.12}$$

$$cosh(x \pm y) = cosh(x)cosh(y) \pm sinh(x)sinh(y), \tag{3.13}$$

$$tanh(x \pm y) = \frac{tanh(x) \pm tanh(y)}{1 \pm tanh(x)tanh(y)}. \tag{3.14}$$

The above observations show that the phase and width parameters within the nonlinear activation functions could be approximated using linear combination of power and product terms. It is possible to use these observations to construct a network model as an extension of the above reduced model which provides an effective linear combination of power and product terms for approximating those nonlinear parameters within the hyperbolic basis function.

On top of the properties observed so far, there are certain activation characteristics which deserve some attention before the function can be chosen as the basis function for the combination. Essentially, the output of each basis function should not be infinitely large at the origin as it gives rise to unstable zero inputs. Also, the output range is preferably free from any value offset which results in possible biased approximation. A plot on these functions shows that the $cosh$ function needs to be offset by -1 in order to have a zero origin. Since $coth$ and $csch$ functions have functional values

at infinity at the origin, they were not included in experiments. The $sech$ function was also excluded as it gives rise to matrices which are close to singular or badly scaled. The remaining hyperbolic function networks under consideration are then labelled as follows [65]:

$$\text{SINH:} \quad \hat{y}_{sinh} = f_{RM}(\boldsymbol{\alpha}, sinh(\boldsymbol{x})), \tag{3.15}$$

$$\text{COSH:} \quad \hat{y}_{cosh} = f_{RM}(\boldsymbol{\alpha}, cosh(\boldsymbol{x}) - 1), \tag{3.16}$$

$$\text{TANH:} \quad \hat{y}_{tanh} = f_{RM}(\boldsymbol{\alpha}, tanh(\boldsymbol{x})). \tag{3.17}$$

3.1.6 Ramp and step networks (RAMP and STEP)

In addition to above hyperbolic functions as seen in [65], in this thesis, two new basis functions, $ramp$ and $step$, are included. They are defined as:

$$ramp(x) = \begin{cases} 1, & x \geqslant 1 \\ x, & -1 < x < 1 \\ -1, & x \leqslant -1 \end{cases}, \tag{3.18}$$

$$step(x) = \begin{cases} 1, & x \geqslant 0 \\ -1, & 0 < x \end{cases}, \tag{3.19}$$

and the corresponding networks are written as:

$$\text{RAMP:} \quad \hat{y}_{ramp} = f_{RM}(\boldsymbol{\alpha}, ramp(\boldsymbol{x})), \tag{3.20}$$

$$\text{STEP:} \quad \hat{y}_{step} = f_{RM}(\boldsymbol{\alpha}, step(\boldsymbol{x})). \tag{3.21}$$

3.2 Experimental Setup

3.2.1 The University of California at Irvine (UCI) data sets

The data sets used in our experiments are all obtained from the UCI Machine Learning Repository [72]. These data sets constitute a large portion of the remaining UCI data which are different from the data sets used in [63]. The purpose of choosing these data sets is to carry out further experiments on the reduced multivariate polynomial model (RM) mentioned above. Different from [63], standardized exhaustive tuning has been performed to all studied classifiers running on the same machine. With these results, it is possible to have a better understanding regarding the behavior of the model and its extensions on a wider range of data sets. The data sets are organized according to the number of classes into three groups: 2-class problems (14 sets), 3-class problems (11 sets) and multi-class problems (6 sets). The purpose of this division is to observe possible trends related to the number of classes. Table 3.1 provides a summary of the data sets used in our experiments including the numbers of classes, attributes, nominal attributes, instances and missing values of each data set. The readers are advised to refer to the web-site [72] for more details.

3.2.2 Performance criteria

Average classification accuracy. In all the experiments, the classification test errors are estimated using 10-fold stratified cross validation and this cross validation is repeated ten times using different random re-ordering of the samples in the data set[1]. The same sets of re-orderings have been used for all classifiers compared. The minimum (min), average (ave), maximum (max) and standard deviation (std) of the classification accuracy (i.e. one minus the error rate) of these ten runs of 10-fold validations are recorded and the average accuracy along with the variance is used as basis for com-

[1]Most reported literatures use a single run of 10-fold cross validation.

Table 3.1: Summary of UCI data sets used

No	Data set name	abbreviation	#instance	#attribute	#nomial	#class	#miss
1	Pittsburgh bridges version 1 TORD	pbri1t	80	7	4	2	28
2	Pittsburgh bridges version 2 TORD	pbri2t	80	7	7	2	28
3	Chess End-Game: King+Rook versus King+Pawn	ckrp	3196	36	36	2	0
4	Chess End-Game: Knight Pin	ckrk	100	16	16	2	0
5	Cylinder bands	cyba	399	23	12	2	142
6	Echocardiogram	echo	61	11	0	2	71
7	Haberman's survival	hasu	306	3	0	2	0
8	Horse colic: surgical	hosl	246	8	8	2	122
9	E. Coli promoter gene sequences	mpgs	106	57	57	2	0
10	Musk database 1	musk1	476	166	0	2	0
11	Musk database 2	musk2	6598	166	0	2	0
12	Spam e-mail database	spam	4601	57	0	2	0
13	SPECT heart	sphe	267	22	0	2	0
14	SPECT heart	sphf	349	44	0	2	0
15	Pittsburgh bridges version 1 MATERIAL	pbri1m	80	7	4	3	28
16	Pittsburgh bridges version 1 SPAN	pbri1s	80	7	4	3	28
17	Pittsburgh bridges version 1 REL-L	pbri1r	80	7	4	3	28
18	Pittsburgh bridges version 2 MATERIAL	pbri2m	80	7	7	3	28
19	Pittsburgh bridges version 2 SPAN	pbri2s	80	7	7	3	28
20	Pittsburgh bridges version 2 REL-L	pbri2r	80	7	7	3	28
21	Horse colic: outcome	hooc	246	8	8	3	122
22	Hayes-Roth	haro	150	3	0	3	0
23	Iris plants	iris	150	4	0	3	0
24	Primate splice-junction gene sequences	msgs	3190	60	60	3	0
25	Postoperative patient	popa	90	8	7	3	3
26	Blocks classification	blcl	5473	10	0	5	0
27	Pittsburgh bridges version 1 TYPE	pbri1y	80	7	4	6	28
28	Pittsburgh bridges version 2 TYPE	pbri2y	80	7	7	6	28
29	Dermatology	derm	358	34	1	6	8
30	Flags database	flag	194	28	18	8	0
31	Cardiac arrhythmia	caar	420	226	40	16	32

parison. This average value is believed to provide a less biased representation of the classifier performance as compared to that from a single run.

Computational efforts. The computing effort is recorded for the training time of the reduced model in terms of standard CPU time unit where each standard time unit is the CPU time taken to evaluate 1000 times the Shekel-5 function at the point (4,4,4,4) [69]. In our experimental setup on a Pentium IV-1.8GHz computer, each standard CPU time unit is equivalent to 0.0569 seconds. Although the standard CPU time unit is machine independence, it nevertheless depends much on efficiency of implementation and computer architecture even using the same machine.[2] The purpose of the standard CPU time unit is to provide some hints about the computing effort for the Matlab implementation under the commonly used Windows environment using the same machine since the difference between the compared algorithms can be up to few hundred times.

Memory storage. The number of learning parameters to be stored for future pattern classification tasks can be an important issue especially for stand-alone applications where only limited memory is available. For model based algorithms (i.e RM and its extensions), the number of weighting parameters to be estimated for the reduced polynomial expansion is tabulated for each data set. For SVM, the parameters are the support vectors and their Lagrange multipliers while for kNN, the parameters are exactly the whole training set. For MLP, the training weights are the parameters to be stored.

3.2.3 Classifier settings

As good training accuracy does not mean good accuracy on unknown data, it is very necessary that the learning algorithms to be well tuned in order to avoid problems like "over-fitting" and "under-fitting". For the algorithms described in section 3.1, this can

[2]Computing resource with vectorization can create much difference among different implementations of matrix multiplications.

be done through tuning the hyper parameters of each algorithm. For RM model and its extensions, the hyper parameter is the degree r. For SVM (using polynomial kernel), the hyper parameters are the degree d of the polynomial kernel, the cost C and the normalization factor γ. For kNN, it is k, the number of nearest neighbors. For MLP, an important hyper parameter is the number of hidden layer nodes. In different problems, good hyper parameters would be different and finding suitable hyper parameters can be done through cross-validation.

In all our experiments, the hyper parameters are found using 10-fold cross-validation as follows. While the test set is kept aside from the procedure, the training set is divided into 10 parts. The division is stratified such that the proportion of classes in all parts is roughly the same. Every part is chosen one-by-one to form a validation set, the rest of corresponding 9 parts are used for training. For each possible value of the hyper parameters, the model (i.e. RM and its extensions, SVM, kNN and MLP) is trained using 9 parts and test on the validation set which gives an accuracy rate. By choosing every part one-by-one to form the validation set, the generalization accuracy can be estimated by averaging the accuracies of those validation sets. The value of hyper parameters which gives best generalization accuracy will be selected for the final 10 runs of test accuracy computation.

In tuning the SVM classifier, d is found from integers between 1 and 10, γ is found from the set $\{0.01, 0.1, 1\}$ and C is found from $\{1, 10, 100\}$. C is chosen from relatively low values because physical data sets are likely to be non-separable and much validation time can be saved from exhaustive search within a small hyper-parameter set. Finding of γ and C in geometric sequences is due to the observation that when theses parameters are small, a small change in the value of these parameters had a large effect on the performance than when they are big [60]. In tuning RM and its extensions, the order r is varied within $[1, 10]$. In tuning kNN, the number of nearest neighbors, k, is varied from integers between 1 and 10. For MLP, let the number of input attributes be l, then the number of hidden nodes in MLP is chosen among the fol-

lowing three values: $l, \lfloor l/2 \rfloor, 2l$. These ranges of search provide a sufficient coverage of possible optimal solution for the compared algorithms.

3.3 Comparison of Classifiers - Experimental Results

3.3.1 CPU time

Table 3.2 lists the standard CPU time needed for training a single fold of the 10-fold cross validation. The time in kNN column is the test time as there is no training in kNN. The table gave some hints regarding the computing speed of different classifiers. The median and the mean training time in standard CPU units of the classifiers are: RM (0.0193, 4.0161), RAMP (0.0209, 5.3867), SINH (0.0228, 5.5553), STEP (0.0288, 39.7847), COSH (0.0424, 19.5415), TANH (0.0524, 8.9203), SVM (0.3639, 168.3742), MLP (42.4712,258.9337) and kNN (0.0176, 32.0237). It is seen that RM, SINH and TANH are faster than the rest in terms of the median and the mean training times among the 31 data sets. The very high average training time of SVM is attributed to the data set `(blcl)` where the best parameters found are $(d, C, \gamma) = (7, 100, 1)$. Without this data set, the mean training time of SVM is only 14.0610 standard CPU units. Among those RM-based classifiers, COSH and STEP are the slowest. This is due to the order of the polynomial needed by these two classifiers being higher than by other RM-based classifiers for some data sets. For example, in the data set `musk2`, the polynomial degree required by COSH and STEP are 4 and 5, respectively whereas other classifiers only require the polynomial degree of 1 or 2. Without `musk2`, the average training time of COSH and STEP are 1.3476 and 3.6750 standard CPU units respectively.

Table 3.2: Running CPU Time (Standard CPU Unit: 1,000 Evaluations of Shekel-5 at (4,4,4,4)).

No.	Dataset	RM	SINH	COSH	TANH	RAMP	STEP	SVM	kNN	MLP
1	pbri1t	0.0105	0.0228	0.0196	0.0000	0.0000	0.0052	0.2016	0.0105	38.8429
2	pbri2t	0.0193	0.0196	0.0000	0.0079	0.0026	0.1073	0.0419	0.0070	38.3298
3	ckrp	8.3197	6.0757	6.6792	3.4921	8.6937	0.6414	28.8168	41.0773	94.0524
4	ckrk	0.0476	0.0467	0.0273	0.0576	0.0524	0.0131	0.0785	0.0264	8.8927
5	cyba	0.2323	0.1400	0.1471	0.1990	0.2277	0.2120	3.3063	0.3040	132.3534
6	echo	0.0053	0.0159	0.0074	0.0524	0.0079	0.0000	0.0209	0.0053	6.9241
7	hasu	0.0211	0.0233	0.0429	0.0288	0.0236	0.0236	4.5864	0.1476	39.8770
8	hosl	0.0000	0.0000	0.3970	0.1387	0.0052	0.1047	5.1309	0.0000	47.4136
9	mpgs	0.0439	0.0467	0.0501	0.0576	0.0497	0.0497	0.1754	0.0633	9.0209
10	musk1	1.2531	1.0112	1.0124	1.1728	1.2513	1.1806	2.6806	4.0492	28.5576
11	musk2	19.3954	97.4340	565.1328	131.7539	130.9241	1123.0759	132.6859	716.5149	79.3168
12	spam	32.3322	21.8568	8.0943	30.9555	12.3115	93.4476	197.8272	124.4077	3644.0707
13	sphe	0.0011	0.0013	0.0016	0.0288	0.0209	0.0288	0.0147	0.0048	101.4110
14	sphf	0.0653	0.0022	0.1460	0.0890	0.0864	0.0890	1.6152	0.0122	143.0916
15	pbri1m	0.0053	0.0074	0.0040	0.0052	0.0079	0.0052	0.0419	0.0123	44.3979
16	pbri1s	0.0404	0.0233	0.0154	0.0026	0.0052	0.0131	0.0785	0.0141	40.6571
17	pbri1r	0.0018	0.0154	0.0037	0.0052	0.0000	0.1178	0.3639	0.0176	41.4346
18	pbri2m	0.0000	0.0000	0.0117	0.0052	0.0079	0.0079	0.0471	0.0141	41.9188
19	pbri2s	0.0035	0.0037	0.0156	0.0000	0.0026	0.0052	0.0812	0.0105	42.4267
20	pbri2r	0.0035	0.0233	0.0040	0.0236	0.0209	0.0000	0.9921	0.0123	42.4712
21	hooc	0.0000	0.0000	0.3970	0.0942	0.0733	0.0079	0.3194	0.1757	71.1152
22	haro	0.0000	0.0000	0.0000	0.0079	0.0157	0.0131	0.2304	0.0000	49.3063
23	iris	0.0105	0.0040	0.0424	0.0157	0.0157	0.0079	0.0628	0.0316	49.3953
24	msgs	42.1845	35.0529	18.9667	101.2513	8.1047	8.7565	36.4293	65.0422	1660.4503
25	popa	0.0123	0.0040	0.0079	0.0000	0.0079	0.0000	0.1152	0.0105	42.3743
26	blcl	17.5975	8.1816	2.0707	4.5157	2.9031	2.9372	4797.7827	36.5624	813.3194
27	pbri1y	0.0088	0.0040	0.0119	0.0000	0.0000	0.0000	0.5497	0.0123	42.0995
28	pbri2y	0.0035	0.0000	0.0077	0.0026	0.0052	0.0026	0.1204	0.0123	41.5759
29	derm	0.0617	0.3022	0.2993	0.3639	0.0707	0.3770	0.4607	0.3199	9.5314
30	flag	0.2076	0.1280	0.1471	0.1597	0.0471	0.0524	0.9660	0.0492	54.0000
31	caar	2.5012	1.7672	1.7990	2.0419	2.0419	2.0419	3.7906	3.8067	528.3168
Median		0.0193	0.0228	0.0424	0.0524	0.0209	0.0288	0.3639	0.0176	42.4712
Mean		4.0125	5.5553	19.5343	8.9203	5.3867	39.7847	168.3747	32.0237	258.9337

3.3.2 Required memory storage

The hyper parameter settings and the number of parameters (p) needed for the nine classifiers are tabulated in Table 3.3 and Table 3.4. Next to the columns of polynomial orders (r) are the ranking (labelled as 'rk') of polynomial order of each classifier compared to other classifiers. The classifier with lowest order is assigned as rank 1 and so on, the classifier with highest order is assigned as rank 7. In cases of ties, an average rank will be assigned for those algorithms which share a similar rank. The purpose of this ranking is to see whether any algorithm persistently needs higher or lower polynomial order (e.g. $rk \rightarrow 1$ or $rk \rightarrow 7$). As can be seen in Table 3.4, RAMP often uses lower order (ave rk = 3.3, ave r = 1.6) while SVM uses higher order than other classifiers (ave rk = 4.8. ave r = 3.0). No classifier is found to persistently use high order.

From Table 3.4, it is easy to see that kNN is the most storage demanding classifier as it has to remember all the training samples provided. Meanwhile, in most of the cases, RM and its extensions using hyperbolic functions use less storage than SVM. However, in some highly nonlinear pattern classification problems which have high input dimension and requires high order approximation, the reduced models are expected to have more weight parameters than those of other models. As shown in Table 3.3, in some data sets (`msgs`, `caar`), RM and its extensions require much more storage than in other data sets. The gain from paying the price of a large number of weight parameters is its single step training that is also least-squares optimal [63]. Also, the number of weight parameters needed by RM and its extensions is seen to be relatively large for high dimensional multi-class problems.

3.3.3 Classification accuracy statistics

The average accuracy of all 9 classifiers over 31 data sets are presented in Table 3.5. In the last row of the table, the means taken across all data sets with respect to each

Table 3.3: Hyper parameter settings and number of parameters of RM, SINH, COSH, TANH classifiers on 31 data sets.

No.	Data set	RM			SINH			COSH			TANH		
		r	rk	p	r	rk	p	r	rk	p	r	rk	p
1	pbri1t	3	5.5	34	3	5.5	34	3	5.5	34	1	1.5	8
2	pbri2t	3	5.5	34	3	5.5	34	1	1	8	2	3	21
3	ckrp	3	4.5	184	3	4.5	184	3	4.5	184	2	2	111
4	ckrk	2	5	51	2	5	51	2	5	51	2	5	51
5	cyba	2	4	72	2	4	72	2	4	72	2	4	72
6	echo	1	3.5	13	1	3.5	13	1	3.5	13	4	7	82
7	hasu	3	3.5	19	3	3.5	19	6	7	40	3	3.5	19
8	hosl	1	2.5	10	1	2.5	10	1	2.5	10	5	6.5	78
9	mpgs	1	3.5	60	1	3.5	60	1	3.5	60	1	3.5	60
10	musk1	1	3.5	170	1	3.5	170	1	3.5	170	1	3.5	170
11	musk2	1	1.5	170	2	4	507	4	6	1181	2	4	507
12	spam	3	5	290	3	5	289	2	2	174	3	5	289
13	sphe	1	3.5	24	1	3.5	24	1	3.5	24	1	3.5	24
14	sphf	3	5	224	1	2.5	46	4	6	313	1	2.5	46
15	pbri1m	1	4	24	1	4	24	1	4	24	1	4	24
16	pbri1s	4	7	141	2	4.5	63	2	4.5	63	1	1.5	24
17	pbri1r	1	2.5	24	2	5	63	1	2.5	24	1	2.5	24
18	pbri2m	1	3.5	24	1	3.5	24	2	7	63	1	3.5	24
19	pbri2s	1	2.5	24	1	2.5	24	2	6	63	1	2.5	24
20	pbri2r	1	2	24	3	5.5	102	1	2	24	3	5.5	102
21	hooc	1	1.5	30	2	3	81	3	4.5	132	4	6.5	183
22	haro	3	4.5	57	2	1.5	36	4	7	78	3	4.5	57
23	iris	2	2.5	45	3	5	72	4	7	99	3	5	72
24	msgs	5	6	1670	4	5	1296	3	4	927	6	7	2034
25	popa	1	3.5	30	1	3.5	30	1	3.5	30	1	3.5	30
26	blcl	8	6.5	795	8	6.5	795	4	3.5	375	4	3.5	375
27	pbri1y	1	3.5	48	1	3.5	48	1	3.5	48	1	3.5	48
28	pbri2y	1	3.5	48	1	3.5	48	1	3.5	48	1	3.5	48
29	derm	1	2	216	2	5.5	630	2	5.5	630	2	5.5	630
30	flag	2	5	696	2	5	696	2	5	696	2	5	696
31	caar	1	3.5	3648	1	3.5	3648	1	3.5	3648	1	3.5	3648
	Mean	2.0	3.9	287.1	2.1	4.1	296.5	2.2	4.3	300.2	2.1	4.0	309.1

r: polynomial order, rk: ranking, p: number of parameters.

classifier is tabulated. For a more detailed accuracy statistics (i.e. in terms of min, max, ave and std) across the 10 runs of 10-fold cross validation for each data set, the readers can refer to Tables A.1, A.2, A.3 in the appendix. In Table 3.5, the bold numbers indicate the classifiers that achieve best accuracy in each data set. It is shown that SVM is the best classifier in 12 data sets. Besides, RM is the classifier that achieves the highest average accuracy among 31 data sets and SVM and TANH follow with small difference. It is also noted that other extensions of RM (SINH, COSH and RAMP nets) also achieve good average accuracy compared to that of SVM (the differences is less than 1%).

Table 3.6 shows the average accuracy of the classifiers with respect to different

Table 3.4: Hyper parameter settings and number of parameters of RAMP, STEP, SVM, KNN, MLP classifiers on 31 data sets.

No.	Data set	RAMP			STEP			SVM			KNN		MLP	
		r	rk	p	r	rk	p	d	rk	p	k	p	nh	p
1	pbri1t	1	1.5	8	2	3	21	3	5.5	161	7	438	3	21
2	pbri2t	2	3	21	7	7	86	2	3	182	5	438	3	21
3	ckrp	3	4.5	184	1	1	38	4	7	9768	4	103608	18	666
4	ckrk	2	5	51	1	1.5	18	1	1.5	986	8	1456	32	544
5	cyba	2	4	72	2	4	72	2	4	5400	3	8280	23	552
6	echo	1	3.5	13	1	3.5	13	1	3.5	96	1	176	5	60
7	hasu	3	3.5	19	5	6	33	2	1	580	10	828	1	4
8	hosl	1	2.5	10	4	5	61	5	6.5	855	9	15.92	4	36
9	mpgs	1	3.5	60	1	3.5	60	3	7	3481	5	5568	29	1711
10	musk1	1	3.5	170	1	3.5	170	2	7	11999	1	72240	84	14196
11	musk2	2	4	507	5	7	1518	1	1.5	49348	1	997752	84	14196
12	spam	2	2	174	5	7	519	2	2	42978	1	236094	57	3306
13	sphe	1	3.5	24	1	3.5	24	2	7	2254	2	5302	22	506
14	sphf	1	2.5	46	1	2.5	46	10	7	4950	1	13860	22	990
15	pbri1m	1	4	24	1	4	24	1	4	210	5	438	6	54
16	pbri1s	1	1.5	24	2	4.5	63	2	4.5	427	5	444	3	27
17	pbri1r	1	2.5	24	8	7	297	6	6	329	7	438	6	54
18	pbri2m	1	3.5	24	1	3.5	24	1	3.5	203	5	438	6	54
19	pbri2s	1	2.5	24	2	6	63	2	6	448	5	444	6	54
20	pbri2r	3	5.5	102	1	2	24	3	5.5	322	4	438	6	54
21	hooc	3	4.5	132	1	1.5	30	4	6.5	1296	2	1592	16	176
22	haro	3	4.5	57	2	1.5	36	3	4.5	276	3	396	6	36
23	iris	3	5	72	1	1	18	2	2.5	190	4	540	8	56
24	msgs	2	2	558	2	2	558	2	2	19654	4	175250	122	7808
25	popa	1	3.5	30	1	3.5	30	2	7	540	10	632	8	88
26	blcl	3	1.5	270	3	1.5	270	7	5	3916	3	49290	10	150
27	pbri1y	1	3.5	48	1	3.5	48	4	7	343	2	438	6	72
28	pbri2y	1	3.5	48	1	3.5	48	2	7	413	2	438	6	72
29	derm	1	2	216	2	5.5	630	1	2	2590	4	11016	17	680
30	flag	1	1.5	240	1	1.5	240	2	5	4379	9	3976	56	2016
31	caar	1	3.5	3648	1	3.5	3648	8	7	70143	4	86784	113	27346
	Mean	1.6	3.3	222.6	2.2	3.7	281.6	3.0	4.8	7700.5	4.4	57388.6	25.4	2438.9

r: polynomial order, rk: ranking, p: number of parameters.

number of classes. It can be seen that, the average accuracy goes down as the number of classes increases for every classifier. Among these 31 UCI data sets, the multi-class problems not only require more training time and memory storage, but also they are harder to be classified than 2-class problems.

3.3.4 Accuracy versus efficiency

One often wants to know the classification accuracy of the classifiers along with their efficiency. In this context, the efficiency refers to CPU time to train a classifier and the amount of memory storage required for parameters. Fig. 3.2(a) plots the average accuracy of each classifier versus its median training time. The median training time is

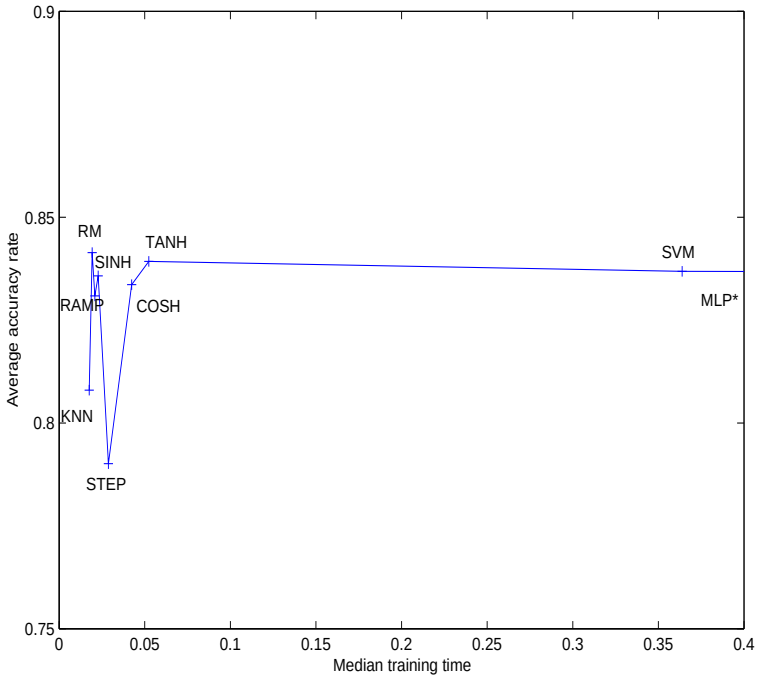
Table 3.5: Classification accuracies of the compared algorithms.

No.	Dataset	RM	SINH net	COSH net	TANH net	RAMP	STEP	SVM	KNN	MLP
1	pbri1t	0.8829	0.8786	0.8700	0.8771	0.8571	0.8671	**0.8986**	0.8514	0.8529
2	pbri2t	0.8529	0.8600	0.8714	0.8586	0.8500	0.8314	0.8757	**0.8943**	0.8571
3	ckrp	0.9450	0.9447	0.9453	0.9440	0.9453	0.9381	**0.9940**	0.9568	0.9920
4	ckrk	0.9756	0.9778	**0.9833**	0.9756	0.9756	0.9411	0.9322	0.8767	0.9344
5	cyba	0.7195	0.7221	0.7369	0.7154	0.7049	0.6410	**0.7587**	0.6708	0.6951
6	echo	0.9140	0.9140	0.9140	0.8640	0.9240	0.9060	0.9800	**1.0000**	0.9820
7	hasu	0.7547	**0.7550**	0.7377	0.7523	0.7520	0.7233	0.7340	0.7217	0.7170
8	hosl	0.8085	0.8085	0.8085	0.8146	0.8167	0.8013	0.8200	0.7234	**0.8308**
9	mpgs	0.9200	0.9120	0.8820	0.9170	0.8890	**1.0000**	0.9140	0.7600	0.8880
10	musk1	0.9415	0.9398	0.9470	0.9476	0.9463	0.9870	**0.9952**	0.9380	0.6948
11	musk2	0.9947	0.9999	0.9814	1.0000	**1.0000**	0.9891	**1.0000**	0.9834	0.8544
12	spam	0.9291	0.9281	0.8137	0.9040	0.7176	0.6281	0.9372	0.9095	**0.9402**
13	sphe	**0.8458**	0.8458	0.8458	0.8458	0.8458	0.8458	0.8254	0.8050	0.8108
14	sphf	0.7718	0.7759	0.7900	0.7779	0.7521	0.7159	**0.8841**	0.8479	0.8171
15	pbri1m	0.9714	0.9714	0.9543	0.9714	0.9700	**0.9829**	0.8571	0.9257	0.9000
16	pbri1s	0.8033	0.7850	0.8017	0.8017	0.8050	0.7900	0.8100	**0.8417**	0.7883
17	pbri1r	0.7086	0.6757	0.7014	**0.7114**	0.7000	0.7100	0.6771	0.7043	0.6814
18	pbri2m	**0.9714**	0.9714	0.9500	0.9714	0.9714	0.9571	0.8571	0.9300	0.8943
19	pbri2s	0.8000	0.7883	0.8067	0.8133	0.8183	0.7967	0.7983	**0.8367**	0.7500
20	pbri2r	0.6986	0.6814	0.7014	0.6886	0.6800	**0.7086**	0.7014	0.6914	0.6557
21	hooc	**0.6809**	0.6809	0.6596	0.6867	0.6496	0.6575	0.6604	0.6596	0.6200
22	haro	0.8571	0.7500	0.8214	0.8687	**0.8673**	0.4187	0.8660	0.6071	0.7587
23	iris	0.9680	0.9653	0.9693	0.9693	**0.9733**	0.7687	0.9640	0.9593	0.9520
24	msgs	**0.9934**	0.9890	0.9909	0.9803	0.9763	0.7907	0.9881	0.8028	0.9912
25	popa	0.7337	0.7325	0.7300	0.7325	0.7325	0.7312	**0.7537**	0.7112	0.6188
26	blcl	0.9574	0.9571	0.9324	0.9554	0.9283	0.9094	0.9701	0.9616	**0.9733**
27	pbri1y	0.6814	0.6786	0.7171	0.6900	**0.7214**	0.6771	0.6186	0.6071	0.6443
28	pbri2y	0.6986	**0.7143**	0.6814	0.6871	0.6986	0.6429	0.5886	0.5957	0.6343
29	derm	0.9721	0.9676	0.9550	0.9709	0.9726	0.9091	**0.9759**	0.9682	0.9729
30	flag	0.5625	0.5738	0.6094	0.5687	0.5475	0.4788	0.5294	**0.6358**	0.5475
31	caar	0.7694	0.7639	0.7333	0.7556	0.7694	0.7500	**0.7778**	0.6703	0.6475
Average		**0.8414**	**0.8358**	**0.8336**	**0.8393**	**0.8309**	**0.7901**	**0.8369**	**0.8080**	**0.8031**
Variance		0.0141	0.0145	0.0124	0.0137	0.0146	0.0216	0.0172	0.0167	0.0179

Table 3.6: Classification accuracy and variance with respect to different number of classes

	RM	SINH net	COSH net	TANH net	RAMP	STEP	SVM	KNN	MLP
2-class	0.8754	0.8759	0.8662	0.8710	0.8555	0.8439	0.8964	0.8528	0.8476
	0.0073	0.0072	0.0068	0.0072	0.0094	0.0161	0.0075	0.0108	0.0095
3-class	0.8453	0.8258	0.8357	0.8463	0.8411	0.7581	0.8180	0.7959	0.7992
	0.0151	0.0170	0.0149	0.0148	0.0165	0.0225	0.0118	0.0145	0.0180
multi-class	0.7679	0.7697	0.7655	0.7657	0.7672	0.7284	0.7449	0.7357	0.7198
	0.0264	0.0248	0.0197	0.0257	0.0246	0.0276	0.0384	0.0311	0.0349

(a)



(b)

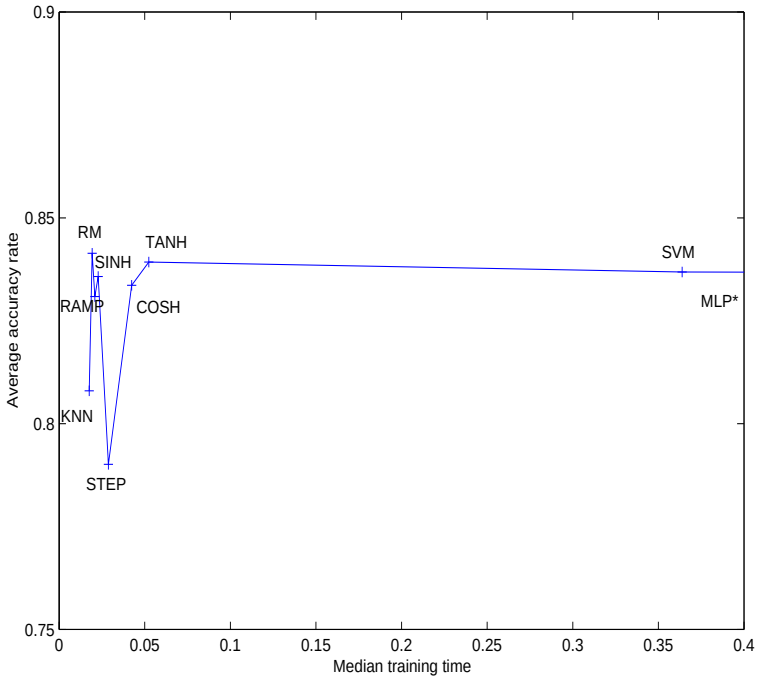


Figure 3.2: (a) Average accuracy versus median training time (in standard CPU unit). For kNN, the test time is included since it requires no training, *: training time of MLP (42.4712) is too high to be displayed, (b) Average accuracy versus average number of parameters

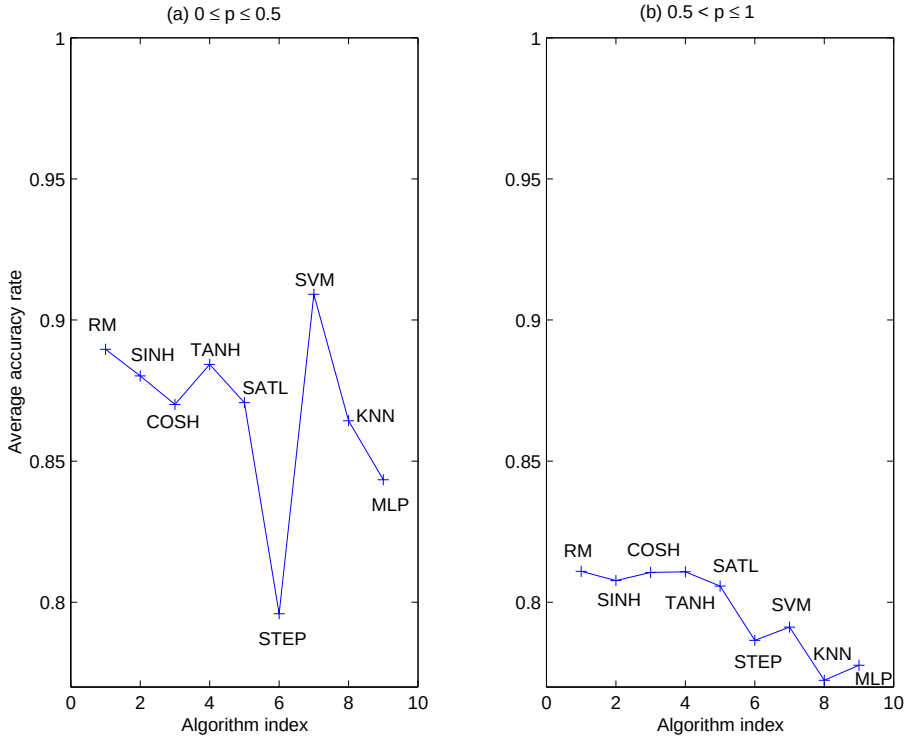


Figure 3.3: Average accuracy according to different proportions of nominal attributes

used instead of the average training time because the average training time is strongly affected by some data sets that require exceptionally long training time (e.g. when SVM is applied to data set `blcl`). It is seen from Fig. 3.2(a) that, although SVM possesses the highest accuracy in many data sets, RM and its extension use much less training time. Fig. 3.2(b) plots the average classification accuracy versus the required memory storage. This figure shows that SVM and kNN requires much more storage than other classifiers while RM and its extensions use roughly similar number of parameters.

3.3.5 Effect of nominal attributes

Another aspect that may affect the classification accuracy of the classifiers is the nominal or categorical attributes. This is because these nominal attributes has to be converted into numerical values before being used in the classifiers. Let p be the propor-

tion of the nominal attributes with respect to the total number of attributes in each data set. For example, the data set `cyba` has 23 attributes and among them 12 are nominal (Table 3.1), thus $p = 12/23 \approx 0.52$. Fig. 3.3 plots the average accuracy over each algorithm (indexed). This figure shows how the classification accuracy of each classifier may be affected by the nominal attributes. Fig. 3.3(a) shows the average accuracy of the classifiers among data sets with p less than 0.5 and Fig. 3.3(b) for p greater than 0.5. These figures show that the studied classifiers have better performance on data sets which have small proportion of nominal attributes (i.e. p less than 50%). Especially, SVM is affected tremendously when the data sets have more nominal attributes (ave accuracy drops from 90.91% down to 79.12%). For STEP, it is seen to be least affected as the accuracy drops only 2%. This shows that although STEP does not possess high accuracy, it is more tolerant to the nominal attributes than other classifiers.

3.3.6 Learning with varying data size and noise

Having chosen the parameters for each classifier using cross-validation, in order to see the robustness of the classifiers, learning data of varying sizes were used. The size of the training data is varied from 25% to 75% of the total available data points and the test data consists of the remaining samples. In addition, noise was added by randomly changing 10% of the class labels of the training samples while keeping the class labels of the test samples unchanged. This is to observe the robustness of the classifiers when they are trained with 10% of the data having wrong class labels. Table 3.7 shows the average classification accuracy on all the 31 data sets for RM, SVM, KNN and MLP. It can be seen from the table that as more samples are added into the training set, a better classification performance is observed. However, the difference between the 50% and 75% columns is not significant. For RM, SVM, KNN, and MLP, the differences are all less than 3%. Besides, as shown in the table, noise does reduce the classification accuracy but not significantly. The deterioration of accuracy due to noise is less than

Table 3.7: Average accuracy with varying learning data size and noise added

Classifiers	25%	25% +	50%	50% +	75%	75% +
RM	0.7320	0.7043	0.7790	0.7535	0.7995	0.7822
SVM	0.7512	0.7294	0.7780	0.7531	0.7982	0.7717
KNN	0.7408	0.7276	0.7649	0.7489	0.7767	0.7568
MLP	0.7207	0.7040	0.7576	0.7324	0.7773	0.7486

x% : without noise, x%+ : with 10% noise added into target class labels

$x = 25, 50, 75$

3% in most cases. Also, when the number of training samples increases (i.e. 50% or more), RM performs slightly better than other algorithms with or without noise added to the class labels.

3.3.7 Summary of results

The above results are summarized in Table 3.8. As seen from the table, SVM has largest best count which is well above all other classifiers. The original RM model possesses the highest average classification accuracy with medium requirement of polynomial degree. Among the hyperbolic extensions of the RM model, TANH has the highest classification performance. RAMP has close performance to TANH but it requires much less polynomial degree than TANH. Combining good accuracy rate, low requirement of memory storage and simple implementation (as can be seen in the Appendix of [63]), the reduced model and its extensions using hyperbolic functions are shown to be good candidates for pattern classification.

3.4 Selection of Classifier for Multiple Biometric Verification

In this chapter, extensive experiments were performed on a reduced multivariate polynomial, its extensions using hyperbolic and nonlinear basis functions, a support vector

Table 3.8: Summary of results for 9 classifiers

Classifiers	Average accuracy	Best count	Average training time	Median training time	Average polynomial degree	Average no. of parameters	Affected by nominal attributes
RM	**0.8414**	4	**4.0125**	**0.0193**	2.0	287.1	$\times\times$
TANH	0.8393	1	8.9203	0.0524	2.1	309.1	$\times\times$
SVM	0.8369	**9**	168.3747	0.3639	3.0	7700.5	$\times\times\times$
SINH	0.8358	2	5.5553	0.0228	2.1	296.5	$\times\times$
COSH	0.8336	1	19.5343	0.0424	2.2	300.2	$\times\times$
RAMP	0.8309	4	5.3867	0.0209	**1.6**	**222.6**	$\times\times$
KNN	0.8080	5	N/A	N/A	N/A	57388.6	$\times\times\times$
STEP	0.7901	3	39.7847	0.0288	2.2	281.6	$\times$
MLP	0.8031	3	258.9337	42.4712	N/A	2438.9	$\times\times$

$\times$: least affected, $\times\times$:medium, $\times\times\times$:most affected

machine, the k-nearest neighbor and a multi-layer perceptron based on 31 data sets from UCI machine learning repository. Ten runs of 10-fold stratified cross validation were performed on these data sets to provide a good understanding regarding the performance statistics of these classifiers. The empirical results show that RM and its extensions are comparable to SVM, MLP and kNN in terms of average accuracy while having significantly faster computing speed. Also, the storage requirement of RM and its extensions is less than those of SVM and kNN. Additionally, the linear formulation of RM is very simple to implement and is possible to be formulated in adaptive form easier than other classifiers. In next chapter, an adaptive updating scheme for multimodal biometric verification which utilizes these characteristics of the RM model will be presented.

Chapter 4

Adaptive Multi-modal Biometrics Fusion

In previous chapter, experimental results have shown that RM model possesses very good classification performance in real world data sets in terms of training time, memory storage and average accuracy rate. These results have led to selection of the RM model as the classifier for multi-modal biometric verification. However, there remain two issues to be resolved when applying the RM model for day-to-day operations. These issues include new user registration and sensor decay problem. In this research, an approach to solve these problems is to formulate the training algorithm in an adaptive fashion. In next section, these problems are discussed in detail. In section 4.2, an adaptive formulation of the RM model is derived. Following are sections 4.3 and 4.4 which discuss the algorithm in different aspects. Finally, a summary section shall conclude the chapter.

4.1 Issues Pertaining to Daily Operation

4.1.1 New user registration

In a security system, new user registration is likely to be a frequent process. As a new user arrives, the system have to be able to record the new identity and to adapt itself so that the new user is able to access the system. In a uni-modal biometric verification system, templates of the new user's biometrics are measured and stored in the database for using in the later matching process. In a multi-modal biometric verification system, not only the biometrics are recorded, but also the combination module has to be updated so that it can recognize the new user.

From the pool of registered users, the training samples consist of the matching score vectors $\boldsymbol{x}_i, i = 1, \ldots, N$ and their labels $y_i, i = 1, \ldots, N$ (i.e. genuine or '1' and impostor or '0'), where N is the number of training samples. From these training samples, the matrix $\boldsymbol{F}_N$ is calculated from the reduced model (3.4) and the optimal parameter $\boldsymbol{\alpha}_N$ is calculated from (3.6). Now, a new user comes and gets registered. Let $\{\boldsymbol{x}, y\}$ be one of these new training samples. The parameter $\boldsymbol{\alpha}_N$ have to be updated so as to adapt the system to the new observations. It is essential that the updating process is fast and, if possible, requires less storage. Otherwise, large database will accumulate over time and this slows down the registration.

The solution for $\boldsymbol{\alpha}$ in (3.6) is a single-step process. This is desirable when the training set is rich, the environment does not change with time, i.e. a static problem. Of course, (3.6) can be used again to update $\boldsymbol{\alpha}$. However, for problems where the training set grows with time, re-training the system using (3.6) might be very time costly. If that is the case, a recursive updating scheme is preferred as in this kind of scheme, the new parameter $\boldsymbol{\alpha}_{N+1}$ is updated using the old parameter $\boldsymbol{\alpha}_N$ and the new training sample $\{\boldsymbol{x}, y\}$ only.

4.1.2 Sequence of biometric data

From results reported in literature [29] and from our own data collection process over a reasonably long period of time, some changes in biometrics data especially the matching scores were noticed. Thus, the biometrics data can be considered as a sequence of data which varies over time. Consider those biometrics that were discussed in chapter 2, the followings are noted:

- **Fingerprint.** Although the biological characteristics of fingerprints may suggest minutia features to be permanent and unchanging for a given finger, acquisition of minutiae information is affected by the skin and imaging conditions at time of measurement and the exact manner the finger was making contact with the sensor. As a result, the measured minutia parameter inevitably changes with time and the measurements can thus be seen as a sequence of data which changes over time. These changes maintain for quite a long time due to the skin nature and human's habits.

- **Speech.** Speech recognition or verification is much affected by noise. It is inevitably that the working condition of the microphone can be very unexpected and often different from the condition in which the speech template of the users are recorded for the first time. Although there are features that are quite insensitive to noise [30], other features may changes dramatically as noise appeared. Besides, sickness, cough and aging do contribute, though slowly, to these changes.

An obvious advantage of recursive learning is that the pattern classifier, starting from some default initialization, is able to improve on the job and to follow changes according to statistical properties of the pattern source. In the context of biometric verification, the pattern sources are the biometric sensors from which data may suffer from noise, decay due to long-term usage or other factors that was discussed above. In such case, a recursive formulation would enhance the verification performance when

combining multi-modal biometrics. In the following section, a recursive formulation will be derived for the RM model.

4.1.3 Recursive learning

Recursive learning (or online learning) is different from batch learning (or off-line learning) in the way it accumulates knowledge. In batch learning, the total training set is required to be available in order to train the systems. Meanwhile, recursive learning changes the accumulated settings (i.e. parameters) of the system whenever new evidence (training sample) becomes available. By this way, the system is updated with every incoming element of the training set.

Online learning is well applied in the field of Neural Networks [13]. For each type of neural networks such as multi layer perceptron (MLP), recurrent networks, radial basis function networks, there is an online learning scheme developed. However, online learning in neural networks only achieves the same solution as that of batch learning in asymptotical manner [13].

Fortunately, if a system uses linear formulation to calculate its parameters, then there exists a recursive learning scheme that will provide the same parameters as the batch learning scheme [58]. At first glance, it may seem that the recursive learning is different from batch learning. Yet, Recursive Least Squares algorithm (RLS) enables us to compute the same parameters as batch learning does with only the knowledge of the new training sample. For example, it is well known that the mean of a random vector $\boldsymbol{\mu} = E\{\boldsymbol{x}\}$ and the moment matrix of that vector $\boldsymbol{M} = E\{\boldsymbol{x}\boldsymbol{x}^T\}$ can be accumulated as more and more samples of $\boldsymbol{x}$ are drawn from its distribution:

$$\boldsymbol{\mu}_n = (1 - \frac{1}{n})\boldsymbol{\mu}_{n-1} + \frac{1}{n}\boldsymbol{x}_n, \tag{4.1}$$

$$\boldsymbol{M}_n = (1 - \frac{1}{n})\boldsymbol{M}_{n-1} + \frac{1}{n}\boldsymbol{x}_n\boldsymbol{x}_n^T. \tag{4.2}$$

The above equations also imply that, not only the mean and the moment matrix can

be accumulated, but any mathematical expression involving them can also be accumulated. At this point, it is necessary to quote J. Schurmann [58]: "Recursive learning is an attractive technique capable of keeping pace with the stream of incoming observations. This feature can be easily combined with pattern classification. The recognition systems works with its already accumulated knowledge and simultaneously improves itself by recursive learning". This is true especially in practice where one can never be sure whether the statistics of the data are stationary or not.

4.2 Recursive Reduced Multivariate Polynomials

4.2.1 Recursive formulation (RM-RLS)

Let $\boldsymbol{f}_i \in \mathcal{R}^K, i = 1, 2, \ldots$ be the row vector of all polynomial terms in (3.4) which is applied to the i-th training sample. All training samples can be packed up to the t-th iteration as:

$$\boldsymbol{F}_t = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{t-1} \\ \boldsymbol{f}_t \end{bmatrix} \text{ and } \boldsymbol{y}_t = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_{t-1} \\ y_t \end{bmatrix}.$$

Let $\boldsymbol{M}_t = \boldsymbol{F}_t^T \boldsymbol{F}_t + b\boldsymbol{I}$, equation (3.6) becomes

$$\boldsymbol{\alpha}_t = \boldsymbol{M}_t^{-1} \boldsymbol{F}_t^T \boldsymbol{y}_t. \tag{4.3}$$

When all training samples are considered equally important, $\boldsymbol{M}_t$ and $\boldsymbol{F}_t^T \boldsymbol{y}_t$ can be rewritten in terms of their past and present instances as follows:

$$\boldsymbol{M}_t = \boldsymbol{F}_t^T \boldsymbol{F}_t + b\boldsymbol{I} = \boldsymbol{F}_{t-1}^T \boldsymbol{F}_{t-1} + \boldsymbol{f}_t^T \boldsymbol{f}_t + b\boldsymbol{I} = \boldsymbol{M}_{t-1} + \boldsymbol{f}_t^T \boldsymbol{f}_t, \tag{4.4}$$

$$\boldsymbol{F}_t^T \boldsymbol{y}_t = \boldsymbol{F}_{t-1}^T \boldsymbol{y}_{t-1} + \boldsymbol{f}_t^T y_t. \tag{4.5}$$

If it is desirable that the system can forget the old training samples, (4.4) and (4.5) can be modified as follows:

$$\boldsymbol{M}_t = (1-\lambda)\boldsymbol{M}_{t-1} + \lambda \boldsymbol{f}_t^T \boldsymbol{f}_t, \tag{4.6}$$

$$\boldsymbol{F}_t^T \boldsymbol{y}_t = (1-\lambda)\boldsymbol{F}_{t-1}^T \boldsymbol{y}_{t-1} + \lambda \boldsymbol{f}_t^T y_t, \tag{4.7}$$

where λ $(0 < \lambda < 1)$ is called the forgetting factor.

Sherman-Morrison-Woodbury matrix inversion formula. In the following matrix manipulation, the Sherman-Morrison-Woodbury matrix inversion formula is used to inverse the matrix $\boldsymbol{M}_t$.

Let the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ satisfy

$$\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}^T \boldsymbol{D} \boldsymbol{C}, \tag{4.8}$$

then the inverse of $\boldsymbol{A}$ is

$$\boldsymbol{A}^{-1} = \boldsymbol{B}^{-1} - \boldsymbol{B}^{-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^T + \boldsymbol{D}^{-1})^{-1}\boldsymbol{C}\boldsymbol{B}^{-1}. \tag{4.9}$$

Apply (4.9) in (4.4) with $\boldsymbol{A} = \boldsymbol{M}_t, \boldsymbol{B} = \boldsymbol{M}_{t-1}, \boldsymbol{C} = \boldsymbol{f}_t, \boldsymbol{D} = 1$, we have

$$\boldsymbol{M}_t^{-1} = \boldsymbol{M}_{t-1}^{-1} - \frac{\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T \boldsymbol{f}_t \boldsymbol{M}_{t-1}^{-1}}{\boldsymbol{f}_t \boldsymbol{M}_{t-1}^{-1} \boldsymbol{f}_t^T + 1}. \tag{4.10}$$

Substitute (4.10) and (4.5) into (4.3), we have

$$\boldsymbol{\alpha}_t = \left[\boldsymbol{M}_{t-1}^{-1} - \frac{\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T \boldsymbol{f}_t \boldsymbol{M}_{t-1}^{-1}}{\boldsymbol{f}_t \boldsymbol{M}_{t-1}^{-1} \boldsymbol{f}_t^T + 1}\right][\boldsymbol{F}_{t-1}^T \boldsymbol{y}_{t-1} + \boldsymbol{f}_t^T y_t]. \tag{4.11}$$

By definition, $\boldsymbol{\alpha}_{t-1} = \boldsymbol{M}_{t-1}^{-1}\boldsymbol{F}_{t-1}^T \boldsymbol{y}_{t-1}$ and with some straightforward matrix manipu-

lations, we finally arrive at

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \frac{\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T(y_t - \boldsymbol{f}_t\boldsymbol{\alpha}_{t-1})}{\boldsymbol{f}_t\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T + 1}. \tag{4.12}$$

By multiplying both sides of (4.10) by $\boldsymbol{f}_t^T$, we have:

$$\boldsymbol{M}_t^{-1}\boldsymbol{f}_t^T = \frac{\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T}{\boldsymbol{f}_t\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T + 1}. \tag{4.13}$$

Substitute the foregoing equation into (4.12), we have a simpler equation:

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \boldsymbol{M}_t^{-1}\boldsymbol{f}_t^T(y_t - \boldsymbol{f}_t\boldsymbol{\alpha}_{t-1}). \tag{4.14}$$

If we use the forgetting factor λ as in (4.6) and (4.7) and follow the similar matrix manipulations, we have:

$$\boldsymbol{M}_t^{-1} = \left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right) - \frac{\left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right)\boldsymbol{f}_t^T\boldsymbol{f}_t\left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right)}{\boldsymbol{f}_t\left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right)\boldsymbol{f}_t^T + \frac{1}{\lambda}}. \tag{4.15}$$

Substitute (4.15) and (4.7) into (4.3), we have:

$$\boldsymbol{\alpha}_t = \left[\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1} - \frac{\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T\boldsymbol{f}_t\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}}{\boldsymbol{f}_t\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T + \frac{1}{\lambda}}\right]\left[(1-\lambda)\boldsymbol{F}_{t-1}^T\boldsymbol{y}_{t-1} + \lambda\boldsymbol{f}_t^T y_t\right], \tag{4.16}$$

which leads to

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \frac{\left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right)\boldsymbol{f}_t^T(y_t - \boldsymbol{f}_t\boldsymbol{\alpha}_{t-1})}{\frac{1}{\lambda}\boldsymbol{f}_t\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T + 1}. \tag{4.17}$$

Multiply both sides of (4.15) by $\boldsymbol{f}_t^T$, we have:

$$\boldsymbol{M}_t^{-1}\boldsymbol{f}_t^T = \frac{\frac{1}{\lambda}\left(\frac{1}{1-\lambda}\boldsymbol{M}_{t-1}^{-1}\right)\boldsymbol{f}_t^T}{\frac{1}{\lambda}\boldsymbol{f}_t\boldsymbol{M}_{t-1}^{-1}\boldsymbol{f}_t^T + 1}. \tag{4.18}$$

Substitute the foregoing equation into (4.17), we have a simpler equation:

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \lambda \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T (y_t - \boldsymbol{f}_t \boldsymbol{\alpha}_{t-1}), \tag{4.19}$$

which is similar to equation (4.14) where no forgetting factor is used.

In (4.12) and (4.17), the new estimate $\boldsymbol{\alpha}_t$ is calculated using the previous estimate $\boldsymbol{\alpha}_{t-1}$, the inversion of $\boldsymbol{M}_{t-1}$ and the new training data $\{\boldsymbol{f}_t, y_t\}$. Thus, these equations are recursive solution for the optimal parameters $\boldsymbol{\alpha}$ in (3.6).

4.2.2 Summary of RM-RLS algorithm

Input: Training set $\boldsymbol{D} = \{\boldsymbol{x}_i, y_i\}, i = 1, 2, \ldots, N$

Output: Parameter vector $\boldsymbol{\alpha}$

1. Initialization: $\boldsymbol{M}_0^{-1} = \frac{1}{b}\boldsymbol{I}, t = 1$, $\boldsymbol{\alpha}_0$ is random.
2. At time t, calculate $\boldsymbol{f}_t$ in (3.4) from $\{\boldsymbol{x}_t, y_t\}$.
3. Update $\boldsymbol{M}_t^{-1}$ and $\boldsymbol{\alpha}_t$ using (4.10) and (4.12) (or (4.15), (4.17))
4. Assign $t \leftarrow t + 1$.

   If $t > N$ then $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_N$ and stop,

   otherwise repeat from step 2.

4.3 An Upper Bound of the Forgetting Factor

The adaptive algorithm, which based on equation (4.19), relies on the forgetting factor λ. If λ is too large, the learning process may not converge and thus, results in undesired solution of the coefficient $\boldsymbol{\alpha}$. In this section, an upper bound for λ which specifies a "safe" range, over which λ can vary, is estimated.

Let Δd_t and Δd_t^* be the errors before and after updating the coefficient $\boldsymbol{\alpha}$ at the t-iteration, we have:

$$\Delta d_t = y_t - \boldsymbol{f}_t \boldsymbol{\alpha}_{t-1}, \tag{4.20}$$

$$\Delta d_t^* = y_t - \boldsymbol{f}_t \boldsymbol{\alpha}_t. \tag{4.21}$$

Substitute (4.19) into (4.21):

$$\begin{aligned} \Delta d_t^* &= y_t - \boldsymbol{f}_t(\boldsymbol{\alpha}_{t-1} + \lambda \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T (y_t - \boldsymbol{f}_t \boldsymbol{\alpha}_{t-1})) \\ &= y_t - \boldsymbol{f}_t^T \boldsymbol{\alpha}_{t-1} - \lambda \boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T (y_t - \boldsymbol{f}_t \boldsymbol{\alpha}_{t-1}) \\ &= \Delta d_t - \lambda \boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T \Delta d_t \\ &= (1 - \lambda \boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T) \Delta d_t. \end{aligned} \tag{4.22}$$

Equation (4.22) shows how the error is updated during each individual iteration of updating the coefficient $\boldsymbol{\alpha}$. For the learning process to converge (see e.g. [58]), the error after updating should be smaller than the error before updating which leads to:

$$1 - \lambda \boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T < 1. \tag{4.23}$$

As a result, the forgetting factor should be smaller than

$$\lambda_{max} = \frac{1}{\boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T}. \tag{4.24}$$

Using the property that $\boldsymbol{a}^T \boldsymbol{b} = trace[\boldsymbol{b}\boldsymbol{a}^T]$, and by taking $\boldsymbol{a}^T = \boldsymbol{f}_t, \boldsymbol{b} = \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T$, we have

$$\boldsymbol{f}_t \boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T = trace[\boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T \boldsymbol{f}_t]. \tag{4.25}$$

From (4.4), as $\boldsymbol{M}_{t-1}$ and $\boldsymbol{f}_t^T \boldsymbol{f}_t$ are positive definite, we have

$$trace[E[\boldsymbol{M}_t]] = trace[E[\boldsymbol{M}_{t-1}]] + trace[E[\boldsymbol{f}_t^T \boldsymbol{f}_t]] > trace[E[\boldsymbol{f}_t^T \boldsymbol{f}_t]], \quad (4.26)$$

which leads to

$$trace[E[\boldsymbol{M}_t]^{-1}] < trace[E[\boldsymbol{f}_t^T \boldsymbol{f}_t]^{-1}] \quad (4.27)$$

and the expectation of the denominator of equation (4.24) is

$$trace[E[\boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T \boldsymbol{f}_t]] < trace[E[\boldsymbol{f}_t^T \boldsymbol{f}_t]^{-1} E[\boldsymbol{f}_t^T \boldsymbol{f}_t]] = trace(I) = K, \quad (4.28)$$

where K is the number of polynomial terms used in equation (3.4).

Thus, a practical limit of λ is:

$$\lambda_{max} = \frac{1}{trace[E[\boldsymbol{M}_t^{-1} \boldsymbol{f}_t^T \boldsymbol{f}_t]]} > \frac{1}{K}. \quad (4.29)$$

Equation (4.29) gives a rough estimation of the limit for the forgetting factor λ. In practice, a value λ smaller than this limit is a sufficient condition for the learning process to converge.

4.4 Remarks and Summary

4.4.1 Remarks on RM-RLS algorithm

- From mathematical point of view, it should be made clear that in the foregoing derivation, no fundamental difference exists between batch estimates in (3.6) and recursive estimates as those given by (4.12) and (4.17).
- The time characteristic of the learning process can be adjusted by proper use of a constant λ, called the forgetting factor. Large λ makes the system forget old training samples faster while small λ allows old training samples to contribute

more in the calculation of $\boldsymbol{\alpha}$.

- With $\boldsymbol{M}_0^{-1}$ being initiated deterministically, RM-RLS requires no matrix inversion like the original RM algorithm. Although RM-RLS still requires the inverse of a scalar, the inverse of $\boldsymbol{M}_t$ in equation (4.10) is exact. Thus, RM-RLS and RM share the numerical stability that regularization method bring about.

- The storage size of RM-RLS consists of of the inversion of $\boldsymbol{M}_t$ ($K \times K$) and the current data ($K \times 1$) which is smaller than the storage size of RM, $(N \times K)$ where N is the number of training samples.

- The RM-RLS can be easily implemented in common programming languages like C or Matlab. In appendix B, an implementation of the algorithm in Matlab is attached.

4.4.2 Summary

In this chapter, two issues in multi-modal biometric verification are discussed. New user registration and sensor decay problems motivate the need for an efficient adaptive updating scheme. Such updating algorithm was derived for the RM model in section 4.2. Without the forgetting factor λ, the RM-RLS algorithm would give the same solution of $\boldsymbol{\alpha}$ as the original RM model. However, for a more flexible learning process, the forgetting factor λ is introduced to indicate how fast the system forgets old training samples. In order to achieve a stable learning process, this forgetting factor should be smaller than a limit given in equation (4.29). RM-RLS also requires no matrix inversion, less memory storage than RM and can be implemented easily in common programming languages. In next chapter, the usefulness of RM-RLS algorithm in multi-modal biometric verification will be demonstrated using various experiments.

Chapter 5

Experimental Results and Discussions

In this chapter, experiments on real data sets are reported to demonstrate that (i) RM-RLS algorithm (see chapter 4) can be applied to speed up new user registration process, and (ii) RM-RLS algorithm is capable of enhancing the verification performance and adapting the biometric verification system to changes in matching scores. The new RM-RLS algorithm shall be evaluated in a few aspects: its verification performance (i.e. accuracy and speed), its robustness to different biometric sensors, and its adaptive characteristics.

In order to carry out the experiments, biometric data sets from different sensors have been collected over a reasonably long period. For the adaptive estimation case, biometric data collected in the first week is used as the reference templates. Biometric data collected in the later weeks are matched with these templates to generate subsequent genuine scores and impostor scores in order to build training and test sets. For consistency in terms of data size, biometric data sets taken from the first two weeks are used in the experiments for static estimation case. It is noted that results on larger data sets have been available in the literature for static estimation case [64–66]. The purpose of carrying out experiments on the static estimation case here is to have a baseline for comparing the case in which the sequence of data varies over time.

Organization of this chapter is as follows: biometric data acquisition and single

biometric verification performance are reported in section 5.1. The verification performance of combining of two and three biometrics is reported in section 5.2. The adaptive characteristics of RM-RLS are investigated in section 5.3. Finally, a discussion on the main findings shall conclude the chapter.

5.1 Single Biometric Verification: Experimental Setup

5.1.1 Fingerprint verification

The representation for fingerprint consists of a global structure and a local structure. The global structure consists of positional and directional information of ridge endings and ridge bifurcations. The local structure consists of relative information of each detected minutia with other neighboring minutiae. Fingerprint verification is then performed by comparing the minutia information between two templates [27]. The interested readers are referred to [27] and [28] for details of minutiae detection and matching.

Data acquisition. In order to observe any change in the matching score over time, the fingerprint data sets were collected over a reasonably long period of time using two type of sensors: Veridicom sensor (20 weeks) and Secugen sensor (30 weeks). Based on visible changes detected in empirical observation, we believe that such changes lies within this time window. The resolution of a fingerprint image obtained by Veridicom sensor and Secugen sensor is 300×300 and 248×292, respectively. Fig. 5.1 and Fig. 5.2 show some fingerprint image samples obtained by these two sensors during the first 10 weeks. In this section, the verification performance of the two sensors, Veridicom sensor which is a CMOS sensor and Secugen sensor which is an optical sensor, are compared.

Veridicom sensor data set. 12 different fingers (left and right thumb, index, middle fingers) from 2 individuals were sampled every week (1 session per week) for 20

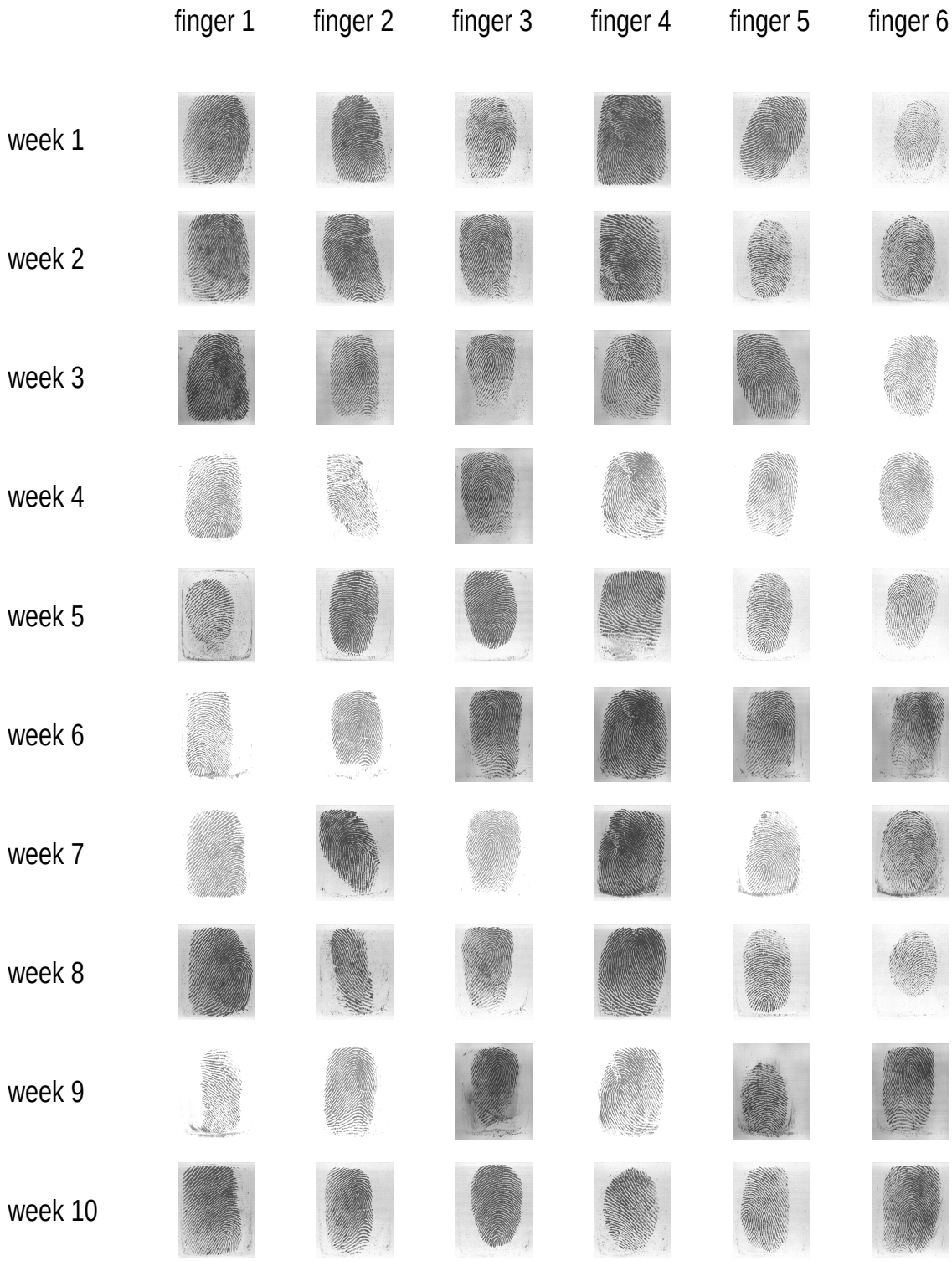


Figure 5.1: Veridicom sensor’s fingerprint image samples

weeks. Since all fingers from the same individual have different fingerprints, we can treat these 12 fingers as those come from 12 different individuals. In each session, 10 samples were collected from each finger. A total of 2400 (12 fingers $\times 10$ samples $\times 20$ weeks) fingerprint images were collected.

Secugen sensor data set. 24 different fingers (left and right thumb, index, middle fingers) from 4 individuals were sampled every week (1 session per week) for 30 weeks. In each session, 10 samples were collected from each finger. A total of 7200 (24 fingers $\times 10$ samples $\times 30$ weeks) fingerprint images were collected.

5.1.2 Speech verification

The speech data set was taken from the commercially available TIDIGIT database [67]. This database consists of speech from both 10 males and 10 females. Each person is required to say digits from 'zero' to 'nine', 10 times each. In this application, the fixed-text mode and the template matching method are adopted for speaker verification [6]. Comparison of two utterances is performed by aligning the two templates at corresponding points in time. To cater for difference in duration of the two utterances, the Dynamic Time Warping (DTW) method is adopted when minimizing a distance metric between two feature sets extracted from the speech data. Fig. 5.3 shows some samples of speech data uttering the word "zero". More details about the system (see also [47, 51] for similar matching designs) can be found in [6].

5.1.3 Hand-geometry verification

In current application, the width and length information are used. First, the hand contour is analyzed and dominant points are located. These points are further identified as finger tips and valleys based on the convex or concave curvature of the contour. The principal axis of each finger is then found by using a set of equally separated grid points starting from respective finger tips. The widths are measured perpendicular to

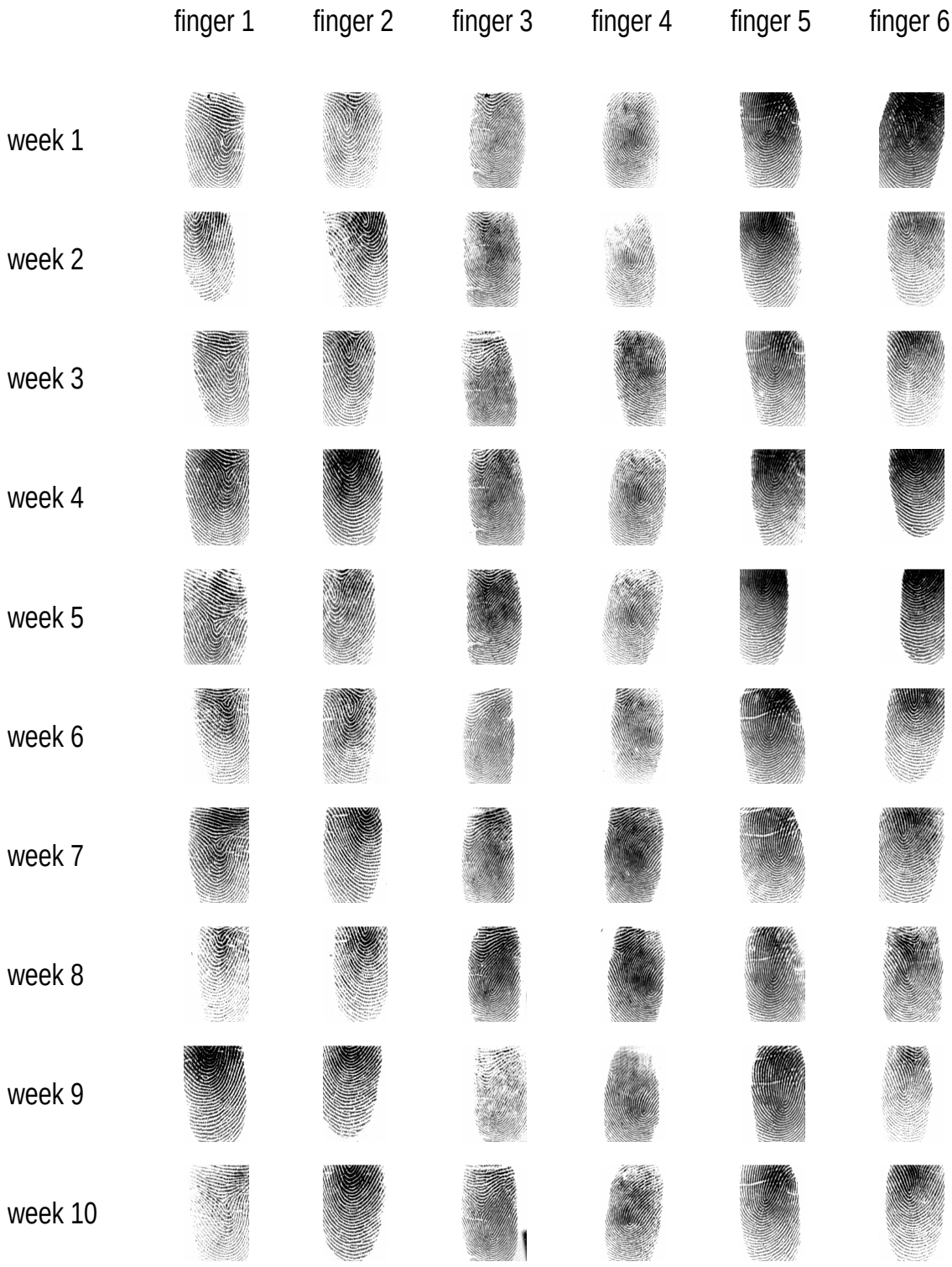


Figure 5.2: Secugen sensor's fingerprint image samples

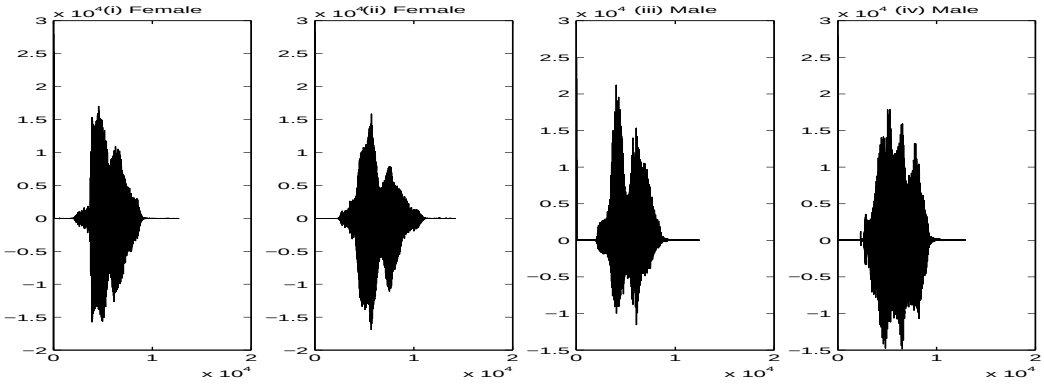
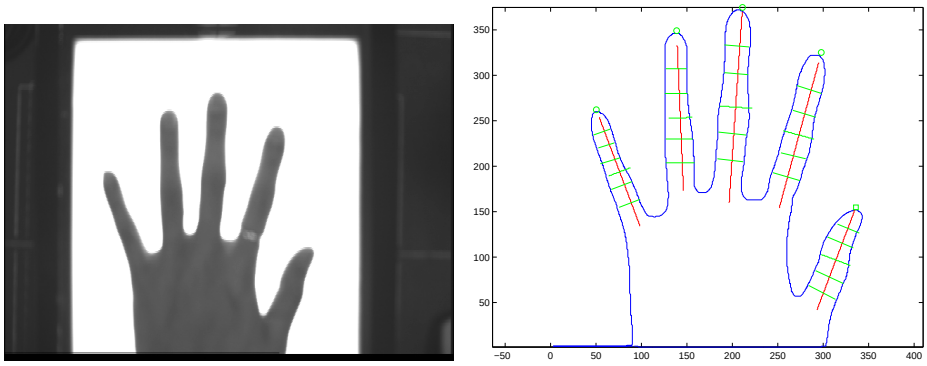


Figure 5.3: Speech samples



(a) (b)

Figure 5.4: (a) A hand image sample, (b) Extracted hand geometry

the axes at the grid points. The features used were similar to those in [26] except that a fixed interval was used for the width measurements. A total of 15 to 30 width features are collected for each hand image depending on the finger length. The length is found using the finger tip and its neighboring valleys information. These features of each finger from both the query image and the template image are compared separately. Their absolute matching differences are summed up and normalized as the matching score. Fig. 5.4 shows a sample captured hand image and its extracted hand-geometry including width/length features in our application. The reader is referred to [68] for more details.

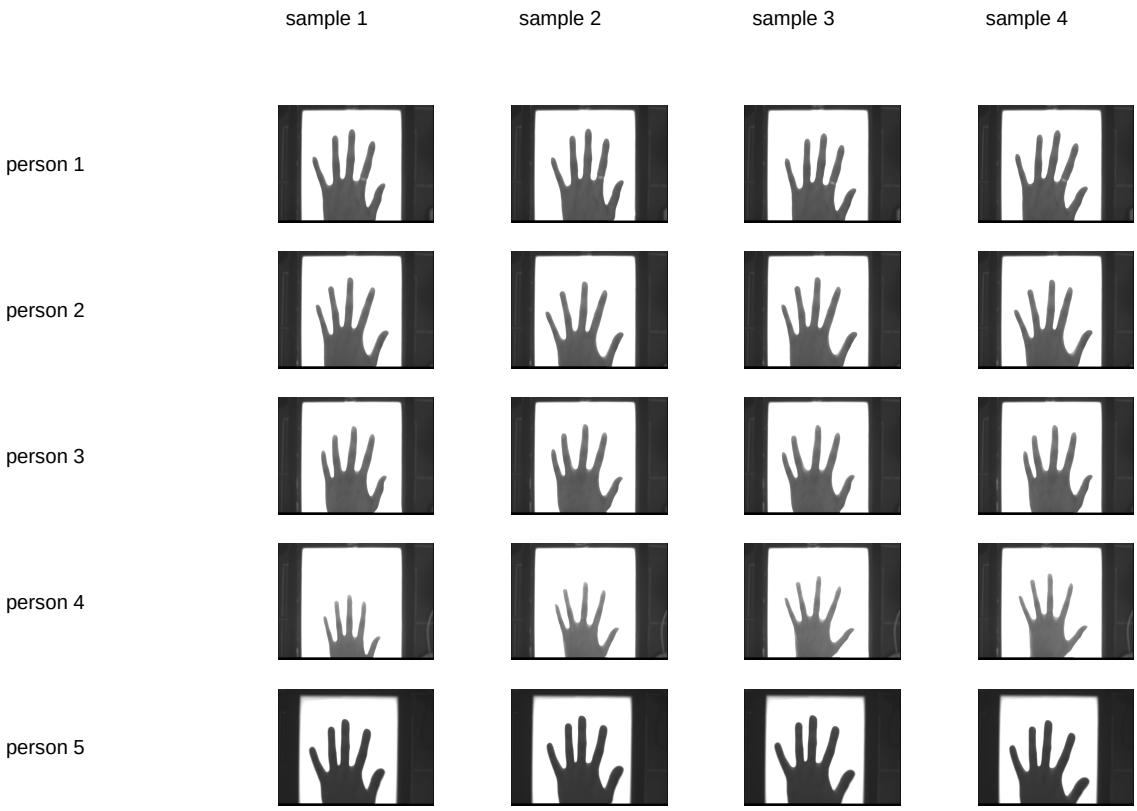


Figure 5.5: Hand image samples

5.1.4 Verification performance

Score normalization. The score is normalized before going on to the final step of combining different biometrics. Reasons behind this normalization step are many. First, scores without normalization can make the final decision biased. Second, scores with high magnitude affect the stability of polynomial-based methods like RM since they may use polynomials with high order. In the following experiments, as there is no assumption on the type of the output function in each biometric matcher as described in [43], a traditional method [23] called *z-score* is adopted. The matching scores are transformed so that they has $zero$ mean value and $unit$ standard deviation. The following normalization steps are performed: first, find the empirical mean m and the standard deviation s; second, subtract the mean from the scores and then divide the scores by the standard deviation:

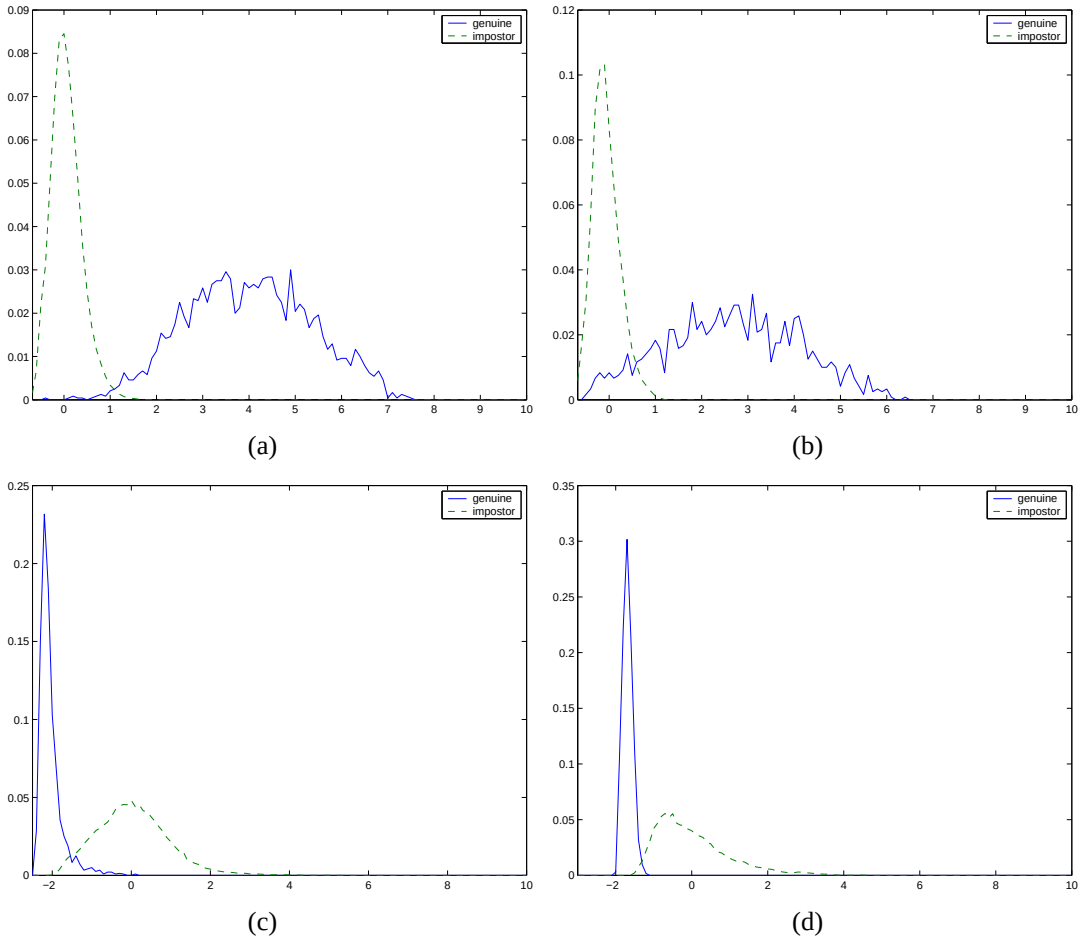


Figure 5.6: Matching scores distributions: (a) Fingerprint (Secugen), (b) Fingerprint (Veridicom), (c) Speech, (d) Hand geometry

$$\text{normalized score} = \frac{\text{score} - m}{s}. \tag{5.1}$$

Score distribution. The matching scores for static fingerprint verification were generated by matching fingerprint samples from the second week with those from the first week. To generate genuine scores, fingerprint samples of the same finger were matched among themselves. To generate impostor scores, fingerprint samples of different fingers were cross matched. Note that there are 12 fingers (representing 12 different identities) in Veridicom dataset and 24 fingers (representing 24 different identities) in Secugen dataset and 10 fingerprint samples were collected for each finger.

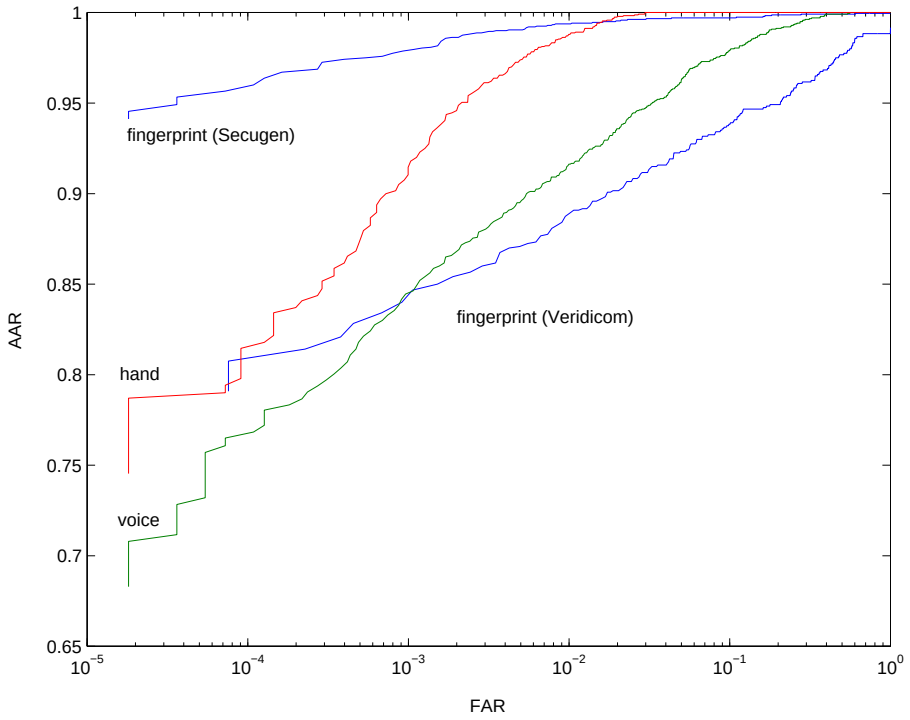


Figure 5.7: ROC curves - single biometric verification

Thus, there are $12 \times 10 \times 10 = 1200$ genuine scores and $12 \times 11 \times 10 \times 10 = 13200$ impostor scores for the Veridicom data set. For the Secugen data set, there are $24 \times 10 \times 10 = 2400$ genuine scores and $24 \times 23 \times 10 \times 10 = 55200$ impostor scores. The genuine scores and impostor scores of speech and hand-geometry verification were generated such that the number of matching scores for each biometric are equal.

Fig. 5.6 shows the matching scores distributions of fingerprint (Veridicom and Secugen), speech and hand geometry for each class of users: genuine users and impostors. The distributions shown were obtained after the normalization step. The figure shows that scores distributions of genuine users and impostors overlap only in the tail (small) area of them. According to the theory of pattern classification [10], this means that for single biometric verification, a simple classification rule based on comparison of the scores with a threshold can be used.

ROC curves. By varying the threshold to various values within the range $[-10, 10]$ (as the scores are now normalized to $zero$ mean and $unit$ standard deviation), for each

biometric, performance criteria like FRR, AAR, FAR are calculated. Fig. 5.7 shows the ROC curves (AAR versus FAR) of each biometric using the entire data set. As can be seen in the figure, fingerprint verification (especially the Secugen data set) has AAR much better than that of speech and hand geometry when the FAR is very small. This means that in a highly secure system, that is, when the threshold is chosen such that a small FAR can be obtained, fingerprint verification with optical fingerprint sensor will be much more reliable than the other two biometrics. The genuine users have higher chance to be accepted by the system. However, it is expected that, combination of many biometrics for verification can have better performance than each biometric alone.

5.2 Multiple Biometric Verification: Experimental Results

In this section, results obtained from two experiments on combination of two and three biometrics for verification are presented. In the first experiment, the data sets are collected using Veridicom sensor, and in the second experiment, the data sets are from Secugen data set. It is noted that, the multi-modal data used in the following experiments are considered "virtual" since the modalities do not come from the same person. The verification performances of RM, SVM (polynomial kernel), and MLP are compared in term of error rates. ROC curves, FRR, EER are used as the performance measures to demonstrate the combination results.

5.2.1 Combination of fingerprint and speech verification

In this experiment, the matching scores are generated using data collected during the first two weeks. This is because the first week collection is considered as the registration process and the second week is considered as day-to-day operation. Fingerprint

images from the same finger are matched to generate genuine matching scores. Fingerprint images from different fingers are matched to generate impostor matching scores. The Veridicom data set consists of 12 different fingers which represent 12 different identities. Thus, the number of genuine matching scores is $1200 (= 12 \times 10 \times 10)$ and the number of impostor matching scores is $13200 (= 12 \times 11 \times 10 \times 10)$. This set of matching scores is divided into two equal parts: training set and test set. The training set was used to train the RM model on different polynomial orders $(r = 1, 2, 3)$. All performance criteria like FAR, FRR, AAR and the ROC curves are computed from the test set.

Fig. 5.8 shows the ROC curves on the test set when combining fingerprint (Veridicom) and speech using different polynomial orders along with the ROC curves of each biometric. The ROC curves of SVM and MLP are also shown in the figure. Note that the ROC curves of SVM and MLP are not as long as that of each biometric as only test set was used instead of the entire data set (i.e. training set and test set). At the operating point $FAR = 0.0001$, the AARs of the RM model of the first, second and third order are 86%, 91%, and 96% respectively while that of fingerprint and speech are 80%, and 78%. As can be seen in the figure, the first order RM model is capable of enhancing the verification performance. The second and third order RM model further enhance the performance.

Table 5.1 shows the FRRs and EERs of RM, SVM and MLP at different settings. It can be seen that RM with 3rd order, SVM with 2nd order and MLP with two hidden nodes perform best among their types of classifiers. SVM with 2nd order performs best with smallest FRR (2.1667%) and smallest EER(0.9848%).

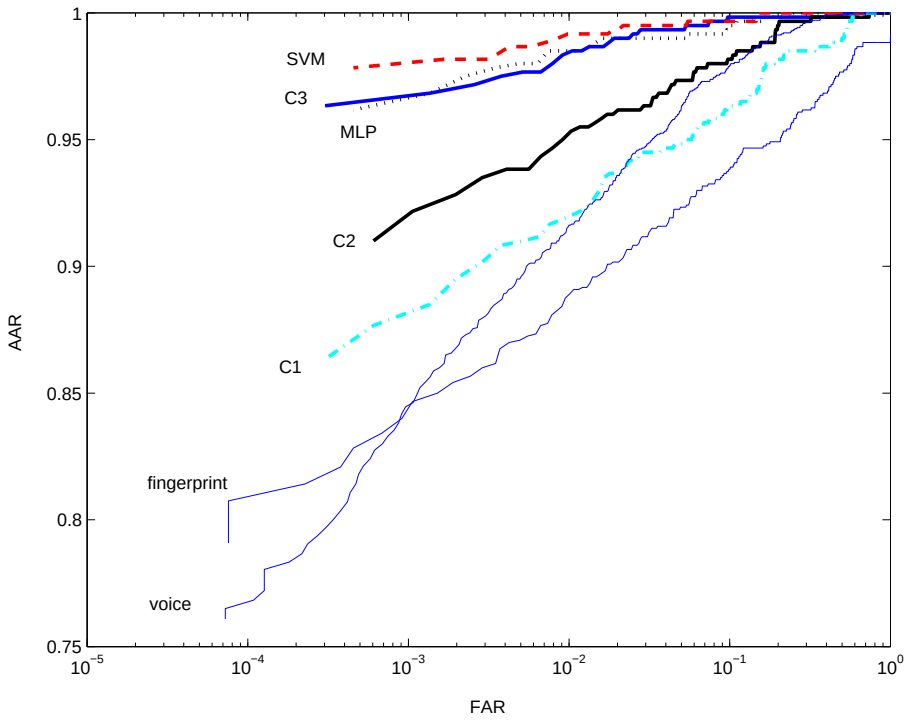


Figure 5.8: ROC curves on test set - combination of fingerprint and speech for verification using Veridicom data set. C1: 1st order RM, C2: 2nd order RM, C3: 3rd order RM.

Table 5.1: Error rates of RM, SVM, MLP - combination of fingerprint and speech.

Classifiers	FRR (%)	EER (%)	AAR (%)
RM ($r = 1$)	13.6667	5.1667	86.3333
RM ($r = 2$)	9.0000	3.3333	91.0000
RM ($r = 3$)	3.6667	1.3333	96.3333
RM ($r = 4$)	5.1667	1.3182	94.8333
RM ($r = 5$)	5.1667	1.3182	94.8333
SVM ($d = 2$)	2.1667	0.9848	97.8333
SVM ($d = 3$)	2.1667	1.5000	97.8333
MLP ($nh = 2$)	3.8333	1.3333	96.1667
MLP ($nh = 3$)	5.1667	1.4848	94.8333

r, d: polynomial order; nh: number of hidden nodes.

5.2.2 Combination of fingerprint, speech and hand-geometry verification

A similar experiment to the above experiment was carried out here. However, the data set collected from Secugen sensor and another biometric which is hand-geometry was used. The Secugen data set consists of 24 different fingers. Thus, the number of genuine matching scores is $2400 (= 24 \times 10 \times 10)$ and the number of impostor matching scores is $55200 (= 24 \times 23 \times 10 \times 10)$.

Fig. 5.9 shows the ROC curves when combining fingerprint (Secugen), speech and hand geometry using different polynomial orders along with the ROC curves of each biometric. Only 5th order RM is shown in the figure as the lines are two close. Table 5.2 shows the FRRs and EERs of RM, SVM and MLP at different settings. It can be seen that RM with 5th order, SVM with 2nd order and MLP with two hidden nodes perform best among their types of classifiers. RM with 5th order performs best with smallest FRR (0.0833%) and smallest EER(0.0362%). At the operating point $FAR = 0.0001$, the AARs of RM models are more than 98% while that of fingerprint, speech and hand-geometry are 94%, 78%, 72%, respectively. As can be seen in the table, the first order RM model is capable of enhancing the verification performance. The higher order RM models (i.e. 2nd, 3rd, 4th, 5th) can further enhance the perfomance.

5.3 Adaptive Multiple Biometric Verification: Experimental Results

5.3.1 Veridicom data set

Fingerpint: As mentioned, 12 fingerprint identities were collected over a period of 20 weeks using Veridicom sensor. For each fingerprint identity, 10 samples were collected weekly. The total number of fingerprints is thus $12 \times 10 \times 20 = 2400$. All fingerprint

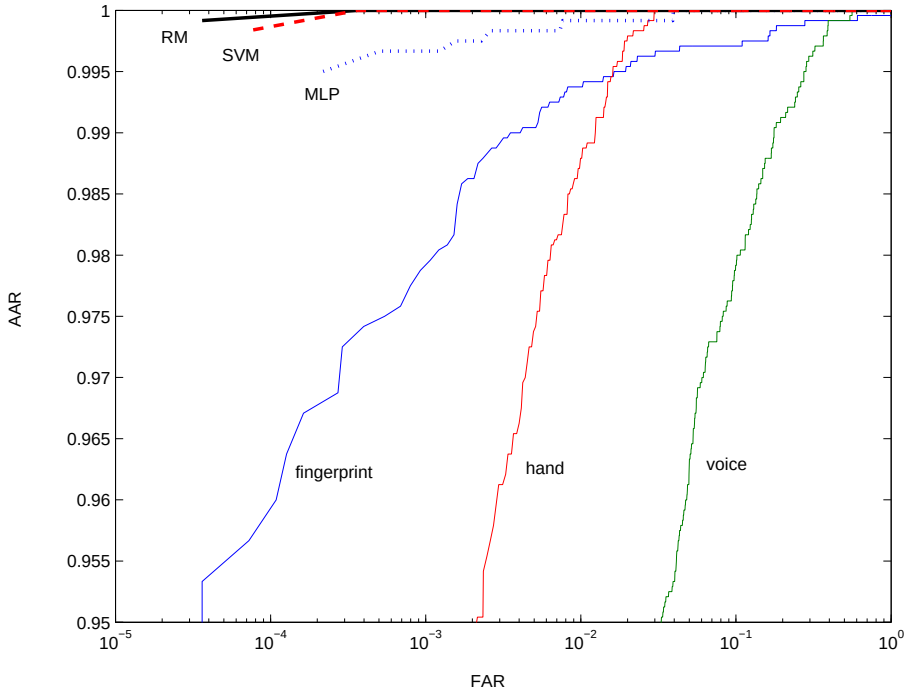


Figure 5.9: ROC curves on test set – combination of fingerprint, speech and hand-geometry for verification using Secugen data set.

Table 5.2: Error rates of RM, SVM, MLP - combination of fingerprint, speech and hand-geometry.

Classifiers	FRR (%)	EER (%)	AAR (%)
RM ($r = 1$)	1.7500	0.4928	98.2500
RM ($r = 2$)	0.4167	0.1522	99.5833
RM ($r = 3$)	0.2500	0.1123	99.7500
RM ($r = 4$)	0.1667	0.0362	99.8333
RM ($r = 5$)	0.0833	0.0362	99.9167
SVM ($d = 2$)	0.1667	0.0362	99.8333
SVM ($d = 3$)	0.1667	0.0362	99.8333
MLP ($nh = 2$)	0.3333	0.1123	99.6667
MLP ($nh = 3$)	0.5000	0.2283	99.5000

r, d: polynomial order; nh: number of hidden nodes.

Table 5.3: CPU times (in sec.) of RM and RM-RLS

No of users	2	4	6	8	10	12
RM	0.01	0.02	0.08	0.30	0.69	1.32
RM-RLS	0.02	0.02	0.04	0.06	0.07	0.08

images collected from the second week and later are matched with the fingerprints collected in the first week to generate the genuine user matching scores and impostor matching scores (see [27] for the minutia matching algorithm). The number of genuine scores generated in each week is $12 \times 10 \times 10 = 1200$. The number of impostor scores generated in each week is $12 \times 11 \times 10 \times 10 = 13200$.

Speech: The spech data was obtained from six persons (3 males and 3 females) taken from TIDIGIT database. Each person was required to say 2 words. Thus, for text-dependent speech verification, there are $6 \times 2 = 12$ identities in total. For each identity, 10 samples were collected. The total number of speech samples is thus $12 \times 10 = 120$. In order to form pairs with the fingerprint identities, a total of 1200 genuine-user matching scores and a total of 13200 impostor matching scores were also generated (see [6] for the speech matching algorithm).

New user registration speed. If the system has N users and a new identity is registered, then the number of genuine user scores and impostor scores added are $\frac{10\times 9}{2} = 45$ and $100 \times N$, respectively. Suppose, at the beginning, the system has no user registered. Each identity is then registered to the system gradually using RM and RM-RLS algorithms. As can be seen in Fig. 5.10 and table 5.3, the CPU time needed to find the final parameter vector $\boldsymbol{\alpha}$ for all 12 identities of RM-RLS (without forgetting) is much less than that of RM as time goes by.

Choice of forgetting factor. In previous section, the RM model with 3rd order ($r = 3$) performs better than the RM model with 1st and 2nd order. Thus, in this experiment, RM-RLS with 3rd order was used. As shown in chapter 4, the forgetting factor should be smaller than a limit $\lambda_{max} \approx \frac{1}{K}$ where K is the number of polynomial

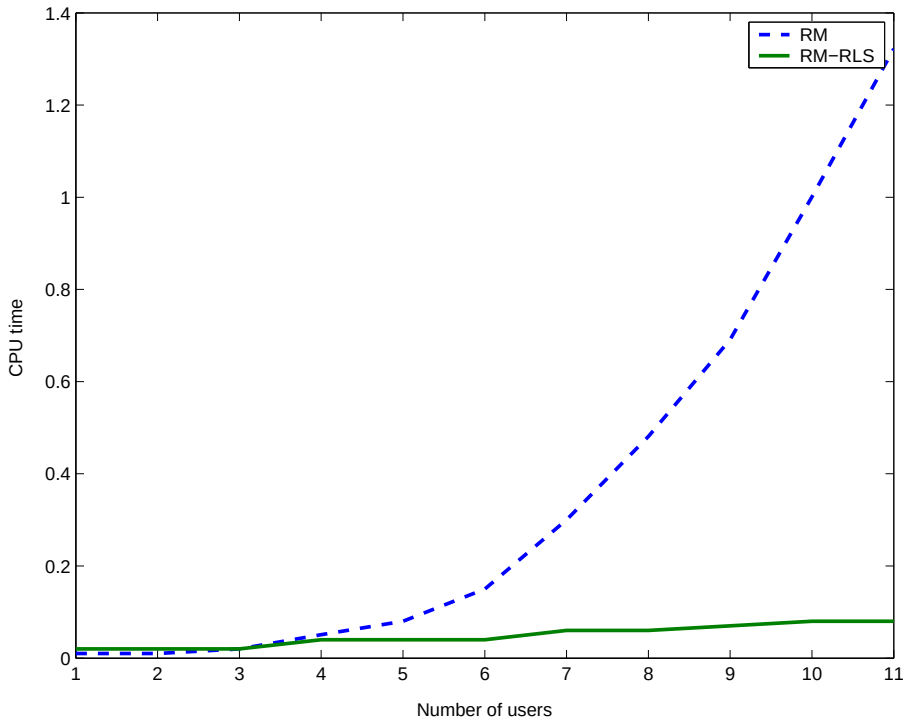


Figure 5.10: CPU times (in sec.) required to find the parameter $\boldsymbol{\alpha}$ of RM and RM-RLS algorithms

terms. With two biometrics ($l = 2$) and 3rd order ($r = 3$), we have $K = 1 + r + l(2r - 1) = 14$. Thus, we have $\lambda_{max} \approx \frac{1}{K} \approx 0.07$. Fig. 5.11 shows the mean squared error of the RM-RLS algorithm over 20 weeks with different λ settings that are smaller than λ_{max}. It can be seen that, with $\lambda = 0.003$ and $\lambda = 0.01$, the mean squared error increases dramatically. As a result, the settings of λ which lie between 0.0003 and 0.003 were used for a stable learning process.

Classification performance. The data set (matching scores) obtained in each week is divided into two equal sets, one for training and the other for test. As the purpose was to show the adaptive capability of the RM-RLS algorithm, the following updating scheme which was used in [29] has been applied. The data from the first week was used to find the parameter $\boldsymbol{\alpha}$ as a initial value. Then in each subsequent week, the current $\boldsymbol{\alpha}$ was used to classify the training set into genuine samples and impostor samples. Only the genuine samples were used to update the parameter vector $\boldsymbol{\alpha}$ using either RM or

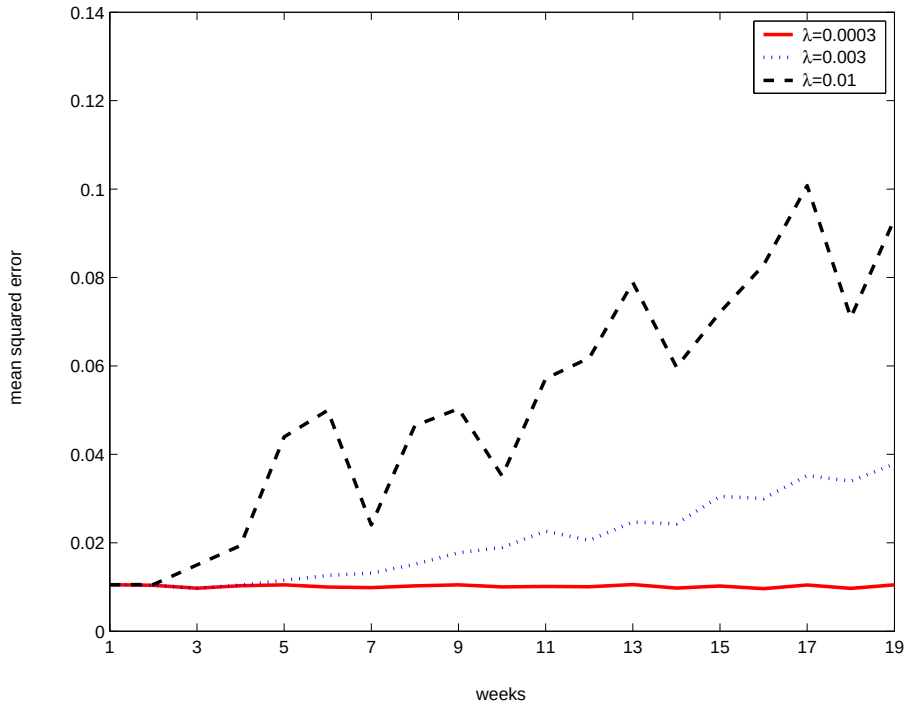


Figure 5.11: Weekly mean squared errors of RM-RLS with different λ settings (Veridicom data set).

RM-RLS (with forgetting factor). For RM-RLS, the following values for the forgetting factor, $\lambda \in \{0.0005, 0.0010\}$ which fall within $[0.0003, 0.003]$, were chosen to see its effect on the training process. Finally, the test sets of all weeks were used to calculate the AARs, the FRRs, the FARs and the ROC curves. The changes of these quantities over time will be observed.

Fig. 5.12 shows the weekly trend of FRR variation for both RM and RM-RLS algorithms at the operating point $FAR = 0.0001$ for 20 weeks. Also, as shown in the figure, in the first few weeks, the performance of RM and that of RM-RLS are similar. From week 10 onwards, RM-RLS with forgetting factor starts to perform better. This shows that there are some changes in the scores. RM-RLS can track these changes and therefore its performance is more steady and better than that of RM (the curves of RM-RLS is below that of RM after week 10).

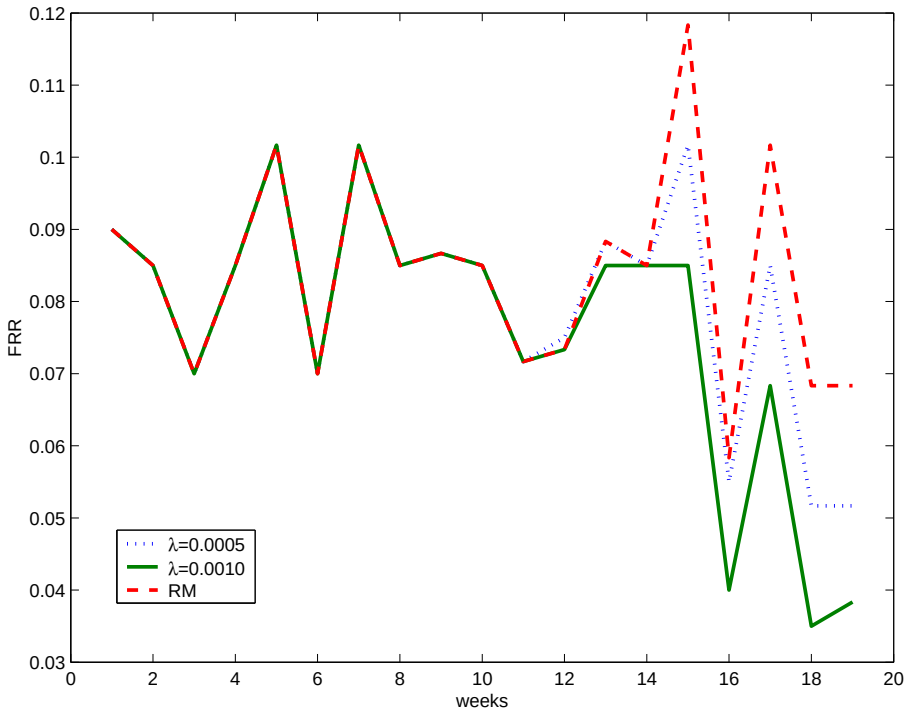


Figure 5.12: FR rates in 20 weeks: combination of fingerprint (Veridicom) and speech.

5.3.2 Secugen data set

Fingerpint: 24 fingerprint identities were collected over a period of 30 weeks using Secugen sensor. For each fingerprint identity, 10 samples were collected weekly. The total number of fingerprints is thus $24 \times 10 \times 30 = 7200$. All fingerprints collected from the second week and later are matched with the fingerprints collected in the first week to generate the genuine user matching scores and impostor matching scores (see [27] for the minutia matching algorithm). The number of genuine scores generated in each week is $24 \times 10 \times 10 = 2400$. The number of impostor scores generated in each week is $24 \times 23 \times 10 \times 10 = 55200$.

Speech: The speech data was obtained from eight people (4 males and 4 females) taken from TIDIGIT database. Each person was required to say 3 words. Thus, for text-dependent speech verification, there were $8 \times 3 = 24$ identities in total. For each identity, 10 samples were collected. The total number of speech samples is thus $24 \times 10 = 240$. In order to form pairs with the fingerprint identities, a total of 2400

genuine-user matching scores and a total of 55200 impostor matching scores were generated.

Hand-geometry: For each hand identity, 10 samples were collected. The total number of hand-geometry samples is thus $24 \times 10 = 240$. In order to form pairs with the fingerprint identities, a total of 2400 genuine-user matching scores and a total of 55200 impostor matching scores were generated.

Choice of forgetting factor. In previous section, the RM model with 5th order ($r = 5$) performs better than than RM model with other orders. Thus, in this experiment, RM-RLS with 5th order was used. With three biometrics ($l = 3$) and 5th order ($r = 3$), we have $K = 1 + r + l(2r - 1) = 33$. Thus, we have $\lambda_{max} \approx \frac{1}{K} \approx 0.03$. Fig. 5.13 shows the mean squared error of RM-RLS algorithm over 20 weeks with different λ settings that are smaller than λ_{max}. It can be seen that, with $\lambda = 0.003$ and $\lambda = 0.01$, the mean squared errors are not stable as that with $\lambda = 0.0003$. As a result, the settings of λ which lie between 0.0003 and 0.003 have been used for a stable learning process.

Classification performance. The same updating scheme described in the previous section was followed here, except that for RM-RLS, the following values for the forgetting factor, $\lambda \in \{0.0003, 0.0005, 0.0007, 0.0009, 0.0010, 0.0011\}$ which fall within $[0.0003, 0.003]$, were chosen to see its effect on the training process. Finally, the test sets of all weeks were used to calculate the AARs, the FRRs, the FARs and the ROC curves. The changes of these quantities over time have been observed.

Fig. 5.14 shows the weekly trend of FRR variation for both RM and RM-RLS algorithms at the operating point $FAR = 0.0001$ for 30 weeks. Only results with $\lambda \in \{0.0005, 0.0010\}$ are shown since lines are too close. As shown in the figure, in the first few weeks, the performance of RM and RM-RLS are similar. From week 10, RM-RLS with forgetting factor starts to perform better. Again, RM-RLS can track the changes in the matching scores and therefore its performance is more steady and better than that of RM (the curves of RM-RLS is below that of RM after week 10).

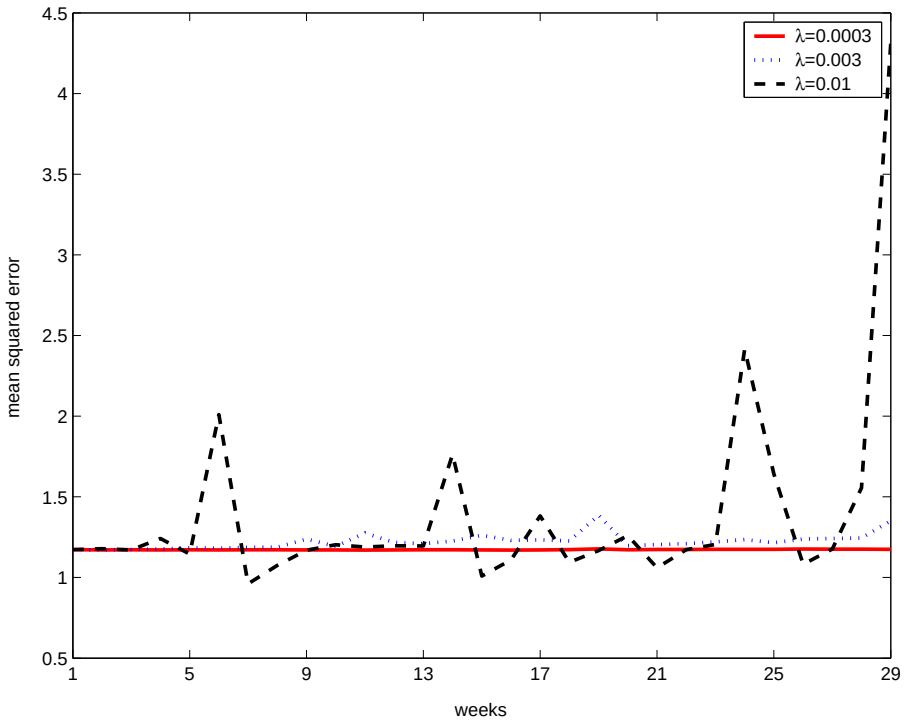


Figure 5.13: Weekly mean squared errors of RM-RLS with different λ settings (Secugen data set).

5.3.3 Data set with artificial noise

This experiment was carried out using the same data set described in the previous section except that noise was added to the speech matching scores. In normal working environment, speech verification may be affected by noise. The noise added has Gaussian distribution with $zero$ mean and the standard deviation (std) which is increased as time goes by. In the first week, std is 0.01 and is added by 0.01 every week. Thus, by the end of 30 weeks, std is 0.3.

Fig. 5.15 shows the weekly trend of FRR variation for both RM and RM-RLS algorithms at the operating point $FAR = 0.0001$ for 30 weeks. As shown in the figure, in the first few weeks, the performance of RM and that RM-RLS are similar. From week 15, the FRR of the RM model starts to increase dramatically while the FRR of RM-RLS with forgetting factor keeps relatively steady. This shows that RM-RLS can track changes in the matching scores, especially when the noise added, and

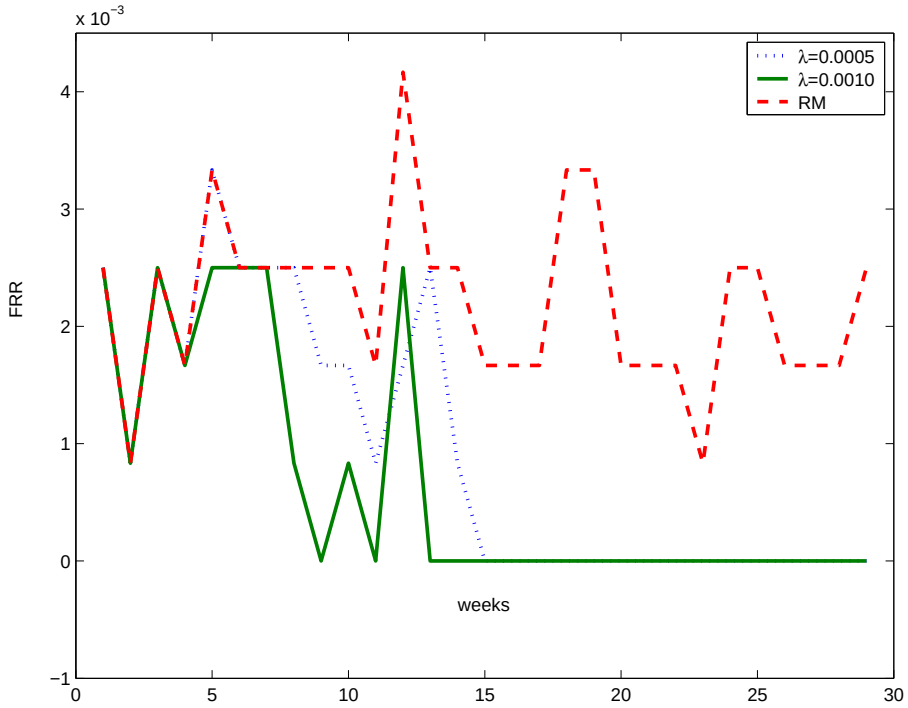


Figure 5.14: FR rates in 30 weeks: combination of fingerprint (Secugen), speech and hand geometry.

therefore its performance is relatively steady and better than that of RM.

5.4 Summary of Results

Main experimental findings in this chapter are:

- **Single biometric verification:** It has been shown that the fingerprint verification performance is better with Secugen (optical) sensor than with Veridicom (CMOS) sensor. According to Fig. 5.7, the ROC curve of Secugen sensor is much higher than that of Veridicom and other two biometrics. It is further noticed that fingerprint images obtained from Secugen sensor is clearer than that from Veridicom sensor. Thus, the minutia information can be extracted more accurately in Secugen sensor than in Veridicom sensor.

- **Speeding up of new user registration process:** Fig. 5.10 shows that RM-RLS

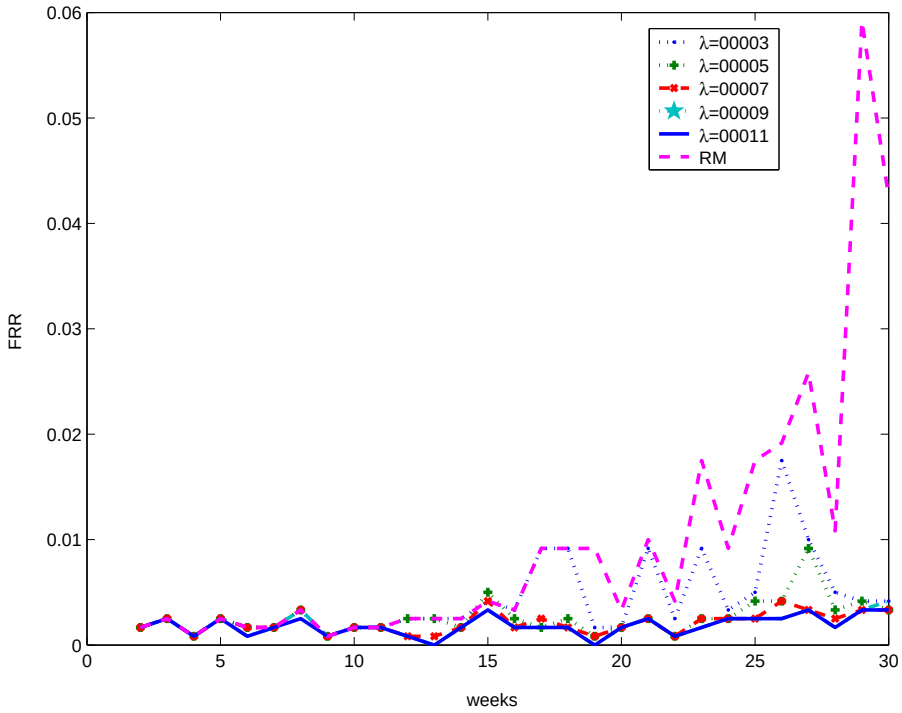


Figure 5.15: FR rates in 30 weeks: combination of fingerprint (Secugen), speech (noise added) and hand geometry.

registration time is being kept steadily as the number of users increases while RM registration time grows very fast. From this result, it can be generalized that when the number of users in the system is huge, the RM-RLS could save up considerable time for computation of learning parameters.

- **Improving verification performance:** Experiments on two data sets (Veridicomm and Secugen) have shown RM-RLS has the ability to keep pace with the incoming matching scores and improve the performance. The FRR of RM-RLS becomes smaller or it is kept more steadily over the later weeks at a reasonable operating point $FAR = 0.0001$. It is concluded that when there may be changes in the score distributions due to sensor decay or noise appeared, RM-RLS can track those changes to maintain and even improve the verification performance of the system.

Chapter 6

Conclusion

Multi-modal biometric verification is gaining more and more attention recently because of the high security level it provides and the non-universality of uni-modal biometrics. Combination of multiple biometrics using classification techniques is an important approach in multi-modal biometric verification. However, for parameterized classifiers, new user registration and adaptation to changes (due to sensor decay or user's habit) could be problematic. New user registration requires retraining the combination module while sensor decay could affect the verification performance. The proposed recursive formulation can solve these problems because (i) it can adapt the combination module efficiently whenever new training samples arrive and (ii) a recursive formulation allows the system to follow changes of statistical properties of the matching scores.

Prior to multi-modal biometrics decision fusion, an empirical comparison of several classifiers including SVM, KNN, MLP, RM and its variants was conducted in this research. Extensive experiments have shown that RM is a good classification tool comparing with other classical techniques like SVM, KNN and MLP. Besides, the single step computation needed to train the RM model allows the solution to be formulated in a recursive fashion. This supported the decision to use the RM model as a basis of the RM-RLS algorithm.

The main focus of this thesis is the development of the RM-RLS algorithm, an advancement of the RM model using Recursive Least Squares method. The proposed algorithm requires only a simple implementation in common programming language like C or Matlab. A short implementation of the algorithm in Matlab is provided in the appendix. It was also shown by experiments that this approach can be very efficient in terms of training time and memory storage needed to find the optimal parameter. The recursive formulation allows the parameters to be accumulated along with new knowledge of incoming training samples instead of being re-calculated using the entire training set.

The RM model and the RM-RLS algorithm have been experimented in multi-modal biometrics decision fusion. The experimental data were collected over a reasonably long period using two types of fingerprint sensors, with and without noise added to the speech data. Three biometrics: fingerprint, speech and hand-geometry were combined for identity verification. The results show that:

1. Multi-modal biometrics decision fusion proposed in this research, using the RM model and RM-RLS algorithm outperforms uni-modal biometric verification. The results are comparable to other classification techniques such as SVM, kNN and MLP.

2. The RM-RLS algorithm has shown the ability to maintain a good performance even when there are changes in the data, such as new user registration.

3. RM-RLS algorithm performs better than RM when there are changes in matching scores due to variations in sensor performance. In addition to using fingerprint data which varies over time, two cases have been considered: (i) without noise and (ii) with noise added to the speech matching scores. In both cases, RM-RLS can maintain or improve (as in case (i)) the verification performance of the system over time.

An immediate challenge in multi-modal biometric verification system is to deal

with the situation where not all biometric measurements are available. This situation may arise in the registration phase as well as in day-to-day operations. The immediate future work would thus be improvement of the RM-RLS algorithm such that it can adapt the verification system in cases where not enough information is available (e.g. missing matching scores). Such algorithm would be very useful in practical multi-modal biometric verification system.

Bibliography

[1] F. M. Alkoot, and J. Kittler. "Experimental Evaluation of Expert Fusion Strategies". *Pattern Recognition Letters*, 20: pp. 1361–1369, 1999.

[2] E. Bauer, and R. Kohavi. "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". *Machine Learning*, 36(1–2): pp. 105–139, 1999.

[3] J. Bigun, B. Duc, F. Smeraldi, S. Fischer, and A. Makarov. "Multi-Modal Person Authentication". *Proceedings Face Recognition: From Theory to Applications*. Nato Advanced Study Institute Programme, 1997.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. New York: Oxford University Press Inc., 1995.

[5] R. Brunelli, and D. Falavigna. "Person Identification Using Multiple Cues". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10): pp. 955–966, 1999.

[6] C. Li and R. Venkateswarlu. "High Accuracy Connected Digits Recognition System with Less Computation". *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, July, Orlando, 2002.

[7] Vassilios Chatzis, Adrian G. Bors, and Ioannis Pitas. "Multimodal Decision-Leval Fusion for Person Authentication". *IEEE Transactions on Systems, Man and Cybernetics*, 29(6): pp. 674–680, 1999.

[8] G. W. Cottrell and M. Fleming. "Face Recognition Using Unsupervised Feature Extraction". *Proceedings of the International Neural Network Conference*, 1: pp. 322–325, 1990.

[9] J. Daughman, "Combining Multiple Biometrics", in *http://www.cl.cam.ac.uk/users/jgd1000/combine/combine.html*.

[10] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc, New York, 2001. (2nd Edition).

[11] S. Dzeroski and B. Zenko. "Is Combining Classifiers Better than Selecting the Best One?", *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 123–130, 2002.

[12] F. Gamble, L. Frye, and D. Grieser. “Real-Time Fingerprint Verification System”. *Applied Optics*, 31(5): pp. 652–655, 1992.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 1999. (2nd Edition).

[14] A. L. Higgins, and J.E. Porter. “Voice Identification Using Nearest-Neighbor Distance Measure”. *Proceeding of the IEEE*, 2: pp. 375–378, 1993.

[15] A. K. Hrechak and J. A. McHugh. “Automatich Fingerprint Recognition Using Structural Matching”. *Pattern Recognition*, 23(8): pp. 893–904, 1990.

[16] T. K. Ho, J. J. Hull, and S. N. Srihari. “Decision Combination in Multiple Classifier Systems”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1): pp. 66–75, 1994.

[17] K. Hornik, M. Stinchcombe, and H. White, “Multi-Layer Feedforward Networks are Universal Approximators”. *Neural Networks*, 2(5): pp. 359–366, 1989.

[18] Y. S. Huang and C. Y. Suen. “A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwriten Numerals”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1): pp. 90–94, 1995.

[19] D. K. Isenor and S. G. Zaky. “Fingerprint Identification Using Graph Matching”. *Pattern Recognition*, 19(2): pp. 113–122, 1986.

[20] S. S. Iyengar, L. Prasad, and H. Min. *Advances in Distributed Sensor Technology*. Prentice Hall, 1995.

[21] A. K. Jain, S. Prabhakar, L. Hong and S. Pankanti. “Filterbank-based Fingerprint Matching”. *IEEE Transactions on Image Processing*, 9(5): pp. 846–859, 2000.

[22] A. K. Jain, L. Hong and R. Bolle. “On-Line Fingerprint Verification”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): pp. 302–313, 1997.

[23] A. K. Jain, K. Nandakumar and A. Ross. “Score Normalization in Multimodal Biometric Systems”. to appear in *Pattern Recognition*, 2005.

[24] A. K. Jain, R. Bolle and S. Pankanti. *Biometrics : Personal Identification in Networked Society*. Kluwer, Boston, c1999.

[25] A. K. Jain and N. Duta. “Deformable Matching of Hand Shapes for User Verification”. *IEEE International Conference on Image Processing*, 1: pp. 857–861, 1999.

[26] A. K. Jain, A. Ross and Sharath Pankanti. “A Prototype Hand Geometry-based Verification System”. *IEEE International Conference on Image Processing*, 1: pp. 857–861, 1999.

[27] X. Jiang and W. -Y. Yau. “Fingerprint Minutiae Matching Based on the Local and Global Structures”. *Proceedings of International Conference on Pattern Recognition*, 2: pp. 1038–1041, 2000.

[28] X. Jiang, W. -Y. Yau and W. Ser. “Detecting the Fingerprint Minutiae by Adaptive Tracing the Gray-Level Ridge”. *Pattern Recognition*, 34(5): pp. 999–1013, 2001.

[29] X. Jiang and W. Ser. “Online Fingerprint Template Improvement”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8): pp. 1121–1126, 2002.

[30] Joseph P. Campbell, Jr.. “Speaker Recognition: A Tutorial”. *Proceedings of the IEEE*, 85(9): pp. 1437–1462, 2000.

[31] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. “On Combining Classifiers”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3): pp. 226–239, 1998.

[32] J. Kittler and K. Messer. “Fusion of Multiple Experts in Multimodal Biometric Personal Identity Verification Systems”. *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, pp. 3–12, 2002.

[33] J. Kittler, and F. M. Alkoot. “Sum versus Vote Fusion in Multiple Classifier Systems”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):pp. 110–115, 2003.

[34] J. Kittler. “A Framework for Classifier Fusion: Is it still needed?”. *Lecture during the joint Statistical Pattern Recognition and SPRR Workshop*, 2000.

[35] U. Kressel, “Pairwise Classification and Support Vector Machines”. *Advances in Kernel Methods - Support Vector Learning* (L. C. Jain and et. al., eds.), ch. 5, (Eds) B. Scholkopf, C. Burges, and A. J. Smola, MIT Press, Cambridge, 1999.

[36] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin. “Decision Templates for Multiple Classifier Fusion: An Experimental Comparsion”. *Pattern Recognition*, 34: pp. 299–314, 2001.

[37] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. W. Duin. “Limits on the Majority Vote Accuracy in Classifier Fusion”. *Pattern Analysis Application*, 6: pp. 22–31, 2003.

[38] L. Lam, and C. Y. Suen. “Application of Majority Voting to Pattern Recognition”. *IEEE Transactions on Systems, Man and Cybernetics - part A*, 27(5): pp. 553–568, 1997.

[39] M. Lades, J Vorbruggen, J. Buhmann, J.Lange, C. V. D. Malburg and R. Wurtz. “Distortion Invariant Object Recognition in The Dynamic Link Architecture”. *IEEE Transactions on Computers*, 42(3):pp. 300–311, 1998.

[40] D. Lee, K. Choi, and J. Kim. "A Robust Fingerprint Matching Algorithm Using Local Alignment". *16 th International Conference on Pattern Recognition (ICPR'02)*, 3: pp. 30803–30806, 2002.

[41] L. Hong and A. K. Jain. "Integrating Faces and Fingerprints for Personal Identification". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): pp. 1295–1307, 1998.

[42] L. Hong, A. K. Jain, and S. Pankanti. "Can Multibiometrics Improve Performance?," in *Proceedings AutoID'99*, pp. 59–64, 1999.

[43] C. L. Liu, H. Hao, and H. Sako. "Confidence Transformation for Combining Classifiers". *Pattern Analysis Application*, 7: pp. 2–17, 2004.

[44] D. Maltoni, Dario Maio, A. K. Jain and Salil Prabhakar. *Handbook of fingerprint recognition*. Springer, 2003.

[45] J. Ma, Y. Zhao, and S. Ahalt, OSU SVM classifier Matlab Toolbox (ver 3.00) in *http://www.eleceng.ohio-state.edu/∼maj/osu-svm/*. Ohio State University, Dept. of Electrical and Computer Engineering.

[46] The MathWorks, Matlab and Simulink, in *http://www.mathworks.com/*, 2005.

[47] J. M. Naik. "Speaker Verification: A Tutorial". *IEEE Communications Magazine*, January: pp. 42–48, 1990.

[48] S. Pankanti, S. Prabhakar, and A. K. Jain. "On the Individuality of Fingerprints". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8): pp. 1010–1024, 2002.

[49] N. Poh Hoon Thian, and S. Bengio. "How do Correlation and Variance of Base-Experts Affect Fusion in Biometric Authentication Task?", to appear in *IEEE Transactions on Signal Processing*, 2005.

[50] S. Prabhakar, and A. K. Jain. "Decision-Level Fusion in Fingerprint Verification". *Pattern Recognition*, 35(4): pp. 861–874, 2002.

[51] L. R. Rabiner. "A Tutorial on Hidden Markove Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE*, 77(2): pp. 257–286, 1989.

[52] Sarunas Raudys. "Experts' Boasting in Trainable Fusion Rules". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9): pp. 1178–1182, 2003.

[53] S. R. Raul, S.A Carmen and G. M. Ana. "Biometric Identification through Hand Geometry Measurements". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10): pp. 1168–1171, 1989.

[54] A. Ross, and A. K. Jain. "Information Fusion in Biometrics". *Pattern Recognition Letters*, 24: pp. 2115–2125, 2003.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Backpropagating Errors". *Nature*, 323(99): pp. 533–536, 1986.

[56] H. Sakoe and S. Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition". *IEEE Transactions on Acoustic, Speech, Signal Processing*, 26(1): pp. 43–49, 1978.

[57] Bernhard Scholkopf and Alex Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[58] Jurgen Schurmann. *Pattern Classification: A Unified View of Statistical and Neural Approaches*. John Wiley & Sons, Inc, New York, 1996.

[59] L. Sha, and X. Tang. "Orientation-Improved Minutiae for Fingerprint Matching". *16 th International Conference on Pattern Recognition (ICPR'04)*, 4: pp. 432–435, 2004.

[60] C. Soares, P. B. Brazdil, and P. Kuba, "A Meta-Learning Method to Select the Kernel Width in Support Vector Regression". *Machine Learning*, 54(3): pp. 195–209, 2004.

[61] F. K. Soong, A.E. Rosenberg, L. R. Rabiner and B. H. Juang. "A Vector Quantization Approach to Speaker Recognition," in Proceeding of the IEEE.

[62] S. Z. Li, D. Zhang, C. Ma, H. Shum, and E. Chang. "Learning to Boost GMM Based Speaker Verification". *Eurospeech, Geneva, Switzerland, 2003*.

[63] K.-A. Toh, Q.-L. Tran, and D. Srinivasan. "Benchmarking a Reduced Multivariate Polynomial Pattern Classifier". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6): pp. 740–755, 2004.

[64] K. A. Toh, W. Y. Yau, and X. Jiang. "A Reduced Multivariate Polynomials Model for Multi-modal Biometrics And Classifiers Fusion". *IEEE Transactions on Circuits and Systems for Video Technology*, 14(2):pp. 224–233, 2004.

[65] K. A. Toh and W. Y. Yau, "Combination of Hyperbolic Functions for Multimodal Biometrics Data Fusion". *IEEE Transactions on Systems, Man and Cybernetics - part B*, 34(2): pp. 1196–1209, 2004.

[66] K. A. Toh, "Fingerprint and Speaker Verification Decisions Fusion," in *Int. Conf. on Image Analysis and Processing, Mantova, Italy, (ICIAP'03)*, pp. 626–631, 2003.

[67] K. A. Toh, W. Y. Yau, and X. Jiang. "Exploiting Global and Local Decisions for Multi-Modal Biometrics Verification". *IEEE Transactions on Signal processing*, 52(10): pp. 3059–3072, 2004.

[68] K. A. Toh, W. Xiong, W. Y. Yau, and X. Jiang. "Combining Fingerprint and Hand-Geometry Verification Decisions," in *4th International Conference on Audio-and Video-Based Biometric Person Authentication, AVBPA'03*, pp. 688–696, 2003.

[69] A. Torn and A. Zilinskas, "Global Optimization," in *Lectures Notes in Compter Science*, Berlin: Springer-Verlag, 1989.

[70] Q.-L. Tran, K.-A. Toh, and D. Srinivasan. "An Empirical Comparison of Nine Pattern Classifiers". *IEEE Transactions on Systems, Man and Cybernetics - part B* (under revision), 2005.

[71] M. A. Turk and A. P. Pentland. "Face Recognition Using Eigen Faces". *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 84–91, 1994.

[72] C. L. Blake and C.J. Merz (1998). UCI Repository of Machine Learning Databases http://www.ics.uci.edu/ mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

[73] V. N. Vapnik, *Statistical Learning Theory*. New York: John Wiley and sons, 1998.

[74] K. Venkataramani and B. V. K. Vijaya Kumar. "Fingerprint Verification Using Correlation Filters". *4th International Conference on Audio-and Video-Based Biometric Person Authentication, AVBPA'03*, pp. 886–894, 2003.

[75] W. P. Kegelmeyer Jr, and K. Bowyer. "Combination of Multiple Classifiers Using Local Accuracy Estimates". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):pp. 405–410, 1997.

[76] S. B. Yacoub, Y. Abdekjaoued, and E. Mayoraz. "Fusion of Face and Speech Data for Person Identity Verification". *IEEE Transactions on Neural Networks*, 10(5): pp. 1065–1074, 1999.

[77] Y. H. Hu, "K Nearest Neighbor Classifier," in *http://www.cae.wisc.edu/∼ece539/matlab/knn.m*. University of Wisconsin-Madison, Computer-Aided Engineering Center.

[78] J. Zhang, Y. Yan, and M. Lades. "Face Recognition: Eigenfaces, Elastic Matching, and Neural Nets, in *Proceedings of the IEEE*, 85(9):pp. 1423–1435, 1997.

Appendix A

Benchmark Experiments on the RM Model

Table A.1: Classification statistics of RM, SINH and COSH

No.	Dataset	RM				SINH net				COSH net			
		min	ave	max	std	min	ave	max	std	min	ave	max	std
1	pbri1t	0.8286	0.8829	0.9286	0.0294	0.8143	0.8786	0.9429	0.0352	0.8429	0.8700	0.9000	0.0207
2	pbri2t	0.8286	0.8529	0.9000	0.0213	0.8286	0.8600	0.9000	0.0221	0.8429	0.8714	0.8857	0.0117
3	ckrp	0.9425	0.9450	0.9465	0.0013	0.9418	0.9447	0.9465	0.0014	0.9428	0.9453	0.9478	0.0017
4	ckrk	0.9556	0.9756	0.9889	0.0102	0.9667	0.9778	0.9889	0.0091	0.9667	0.9833	1.0000	0.0108
5	cyba	0.7103	0.7195	0.7333	0.0068	0.7128	0.7221	0.7333	0.0071	0.7231	0.7369	0.7538	0.0104
6	echo	0.9000	0.9140	0.9400	0.0135	0.9000	0.9140	0.9400	0.0135	0.9000	0.9140	0.9400	0.0135
7	hasu	0.7367	0.7547	0.7667	0.0089	0.7433	0.7550	0.7667	0.0081	0.7233	0.7377	0.7500	0.0096
8	hosl	0.8085	0.8085	0.8085	0.0000	0.8085	0.8085	0.8085	0.0000	0.8085	0.8085	0.8085	0.0000
9	mpgs	0.9000	0.9200	0.9500	0.0156	0.8900	0.9120	0.9300	0.0155	0.8400	0.8820	0.9300	0.0266
10	musk1	0.9348	0.9415	0.9478	0.0044	0.9348	0.9398	0.9435	0.0025	0.9283	0.9470	0.9609	0.0097
11	musk2	0.9945	0.9947	0.9950	0.0002	0.9997	0.9999	1.0000	0.0001	0.9150	0.9814	1.0000	0.0352
12	spam	0.9264	0.9291	0.9314	0.0015	0.9261	0.9281	0.9298	0.0011	0.8111	0.8137	0.8163	0.0014
13	sphe	0.8346	0.8458	0.8538	0.0078	0.8346	0.8458	0.8538	0.0078	0.8346	0.8458	0.8538	0.0078
14	sphf	0.7235	0.7718	0.8000	0.0203	0.7676	0.7759	0.7882	0.0060	0.7618	0.7900	0.8176	0.0205
15	pbri1m	0.9714	0.9714	0.9714	0.0000	0.9714	0.9714	0.9714	0.0000	0.9429	0.9543	0.9714	0.0090
16	pbri1s	0.7667	0.8033	0.8500	0.0281	0.7500	0.7850	0.8167	0.0228	0.7667	0.8017	0.8500	0.0266
17	pbri1r	0.6714	0.7086	0.7429	0.0254	0.6143	0.6757	0.7143	0.0323	0.6714	0.7014	0.7429	0.0273
18	pbri2m	0.9714	0.9714	0.9714	0.0000	0.9714	0.9714	0.9714	0.0000	0.9286	0.9500	0.9714	0.0121
19	pbri2s	0.7667	0.8000	0.8333	0.0222	0.7500	0.7883	0.8167	0.0209	0.7667	0.8067	0.8333	0.0263
20	pbri2r	0.6571	0.6986	0.7286	0.0218	0.6143	0.6814	0.7429	0.0381	0.6857	0.7014	0.7286	0.0142
21	hooc	0.6809	0.6809	0.6809	0.0000	0.6809	0.6809	0.6809	0.0000	0.6596	0.6596	0.6596	0.0000
22	haro	0.8571	0.8571	0.8571	0.0000	0.7500	0.7500	0.7500	0.0000	0.8214	0.8214	0.8214	0.0000
23	iris	0.9667	0.9680	0.9733	0.0028	0.9600	0.9653	0.9733	0.0042	0.9533	0.9693	0.9733	0.0064
24	msgs	0.9909	0.9934	0.9946	0.0012	0.9826	0.9890	0.9918	0.0029	0.9849	0.9909	0.9924	0.0022
25	popa	0.7125	0.7337	0.7500	0.0119	0.7125	0.7325	0.7500	0.0134	0.7125	0.7300	0.7500	0.0105
26	blcl	0.9555	0.9574	0.9590	0.0008	0.9559	0.9571	0.9583	0.0008	0.9289	0.9324	0.9338	0.0015
27	pbri1y	0.6571	0.6814	0.7143	0.0166	0.6571	0.6786	0.7000	0.0154	0.6857	0.7171	0.7429	0.0176
28	pbri2y	0.6714	0.6986	0.7429	0.0218	0.6857	0.7143	0.7429	0.0190	0.6571	0.6814	0.7000	0.0151
29	derm	0.9618	0.9721	0.9794	0.0044	0.9647	0.9676	0.9765	0.0034	0.9441	0.9550	0.9618	0.0061
30	flag	0.5313	0.5625	0.6250	0.0257	0.5500	0.5738	0.6125	0.0228	0.5687	0.6094	0.6375	0.0205
31	caar	0.7558	0.7694	0.7789	0.0124	0.7517	0.7639	0.7622	0.0108	0.7228	0.7333	0.7438	0.0159
Average		0.8248	**0.8414**	0.8595	0.0108	0.8191	**0.8358**	0.8517	0.0108	0.8143	**0.8336**	0.8509	0.0126

Table A.2: Classification statistics of TANH, RAMP and STEP

No.	Dataset	TANH				RAMP				STEP			
		min	ave	max	std	min	ave	max	std	min	ave	max	std
1	pbri1t	0.8429	0.8771	0.9143	0.0215	0.8286	0.8571	0.8857	0.0213	0.8286	0.8671	0.8857	0.0179
2	pbri2t	0.8143	0.8586	0.8857	0.0207	0.8143	0.8500	0.8857	0.0181	0.7857	0.8314	0.8857	0.0322
3	ckrp	0.9425	0.9440	0.9459	0.0011	0.9440	0.9453	0.9472	0.0009	0.9365	0.9381	0.9403	0.0013
4	ckrk	0.9556	0.9756	0.9889	0.0102	0.9556	0.9756	0.9889	0.0102	0.9333	0.9411	0.9556	0.0075
5	cyba	0.7051	0.7154	0.7308	0.0076	0.6821	0.7049	0.7256	0.0135	0.6256	0.6410	0.6667	0.0114
6	echo	0.8000	0.8640	0.9400	0.0450	0.9200	0.9240	0.9400	0.0084	0.8800	0.9060	0.9200	0.0135
7	hasu	0.7333	0.7523	0.7667	0.0089	0.7367	0.7520	0.7600	0.0074	0.7233	0.7233	0.7233	0.0000
8	hosl	0.7917	0.8146	0.8250	0.0125	0.8083	0.8167	0.8292	0.0059	0.7875	0.8013	0.8250	0.0106
9	mpgs	0.8900	0.9170	0.9500	0.0206	0.8600	0.8890	0.9200	0.0223	1.0000	1.0000	1.0000	0.0000
10	musk1	0.9391	0.9476	0.9609	0.0067	0.9391	0.9463	0.9565	0.0057	0.9848	0.9870	0.9891	0.0010
11	musk2	0.9997	1.0000	1.0000	0.0001	1.0000	1.0000	1.0000	0.0000	0.9879	0.9891	0.9898	0.0007
12	spam	0.8357	0.9040	0.9222	0.0294	0.7163	0.7176	0.7198	0.0010	0.6277	0.6281	0.6290	0.0005
13	sphe	0.8346	0.8458	0.8538	0.0078	0.8346	0.8458	0.8538	0.0078	0.8346	0.8458	0.8538	0.0078
14	sphf	0.7647	0.7779	0.8000	0.0110	0.7353	0.7521	0.7706	0.0103	0.7088	0.7159	0.7235	0.0046
15	pbri1m	0.9714	0.9714	0.9714	0.0000	0.9571	0.9700	0.9714	0.0045	0.9714	0.9829	0.9857	0.0060
16	pbri1s	0.7667	0.8017	0.8500	0.0319	0.7667	0.8050	0.8667	0.0334	0.7500	0.7900	0.8333	0.0274
17	pbri1r	0.6857	0.7114	0.7429	0.0200	0.6857	0.7000	0.7143	0.0151	0.6571	0.7100	0.7714	0.0344
18	pbri2m	0.9714	0.9714	0.9714	0.0000	0.9714	0.9714	0.9714	0.0000	0.9429	0.9571	0.9714	0.0067
19	pbri2s	0.7833	0.8133	0.8500	0.0233	0.7833	0.8183	0.8500	0.0254	0.7667	0.7967	0.8333	0.0219
20	pbri2r	0.6429	0.6886	0.7286	0.0250	0.6429	0.6800	0.7143	0.0204	0.6857	0.7086	0.7429	0.0204
21	hooc	0.6583	0.6867	0.7208	0.0211	0.6208	0.6496	0.6667	0.0155	0.6500	0.6575	0.6667	0.0073
22	haro	0.8467	0.8687	0.8933	0.0144	0.8533	0.8673	0.8867	0.0111	0.3800	0.4187	0.4533	0.0247
23	iris	0.9667	0.9693	0.9733	0.0034	0.9667	0.9733	0.9800	0.0044	0.7400	0.7687	0.7867	0.0122
24	msgs	0.9498	0.9803	0.9934	0.0141	0.9751	0.9763	0.9776	0.0008	0.7861	0.7907	0.7950	0.0029
25	popa	0.7125	0.7325	0.7500	0.0134	0.7125	0.7325	0.7500	0.0134	0.7250	0.7312	0.7500	0.0088
26	blcl	0.9540	0.9554	0.9575	0.0012	0.9278	0.9283	0.9287	0.0003	0.9088	0.9094	0.9101	0.0004
27	pbri1y	0.6714	0.6900	0.7143	0.0136	0.6857	0.7214	0.7429	0.0181	0.6714	0.6771	0.6857	0.0074
28	pbri2y	0.6714	0.6871	0.7000	0.0105	0.6857	0.6986	0.7143	0.0125	0.6143	0.6429	0.6714	0.0178
29	derm	0.9647	0.9709	0.9765	0.0040	0.9647	0.9726	0.9794	0.0037	0.9000	0.9091	0.9176	0.0063
30	flag	0.5313	0.5687	0.6312	0.0262	0.5125	0.5475	0.5687	0.0200	0.4313	0.4788	0.5125	0.0236
31	caar	0.7489	0.7556	0.7622	0.0101	0.7589	0.7694	0.7722	0.0101	0.7389	0.7500	0.7622	0.0101
Average		0.8176	**0.8393**	0.8604	0.0140	0.8144	**0.8309**	0.8464	0.0110	0.7730	**0.7901**	0.8076	0.0112

Table A.3: Classification statistics of SVM, KNN and MLP

No.	Dataset	SVM				KNN				MLP			
		min	ave	max	std	min	ave	max	std	min	ave	max	std
1	pbri1t	0.8571	0.8986	0.9429	0.0265	0.8429	0.8514	0.8714	0.0100	0.8286	0.8529	0.9000	0.0213
2	pbri2t	0.8429	0.8757	0.9000	0.0191	0.8714	0.8943	0.9143	0.0138	0.8143	0.8571	0.9000	0.0278
3	ckrp	0.9918	0.9940	0.9959	0.0011	0.9535	0.9568	0.9597	0.0018	0.9906	0.9920	0.9940	0.0011
4	ckrk	0.9111	0.9322	0.9444	0.0110	0.8556	0.8767	0.8889	0.0122	0.9222	0.9344	0.9556	0.0110
5	cyba	0.7385	0.7587	0.7795	0.0135	0.6410	0.6708	0.6974	0.0186	0.6667	0.6951	0.7179	0.0177
6	echo	0.9800	0.9800	0.9800	0.0000	1.0000	1.0000	1.0000	0.0000	0.9400	0.9820	1.0000	0.0220
7	hasu	0.7233	0.7340	0.7433	0.0060	0.7033	0.7217	0.7300	0.0089	0.7067	0.7170	0.7333	0.0090
8	hosl	0.8000	0.8200	0.8333	0.0094	0.7234	0.7234	0.7234	0.0000	0.8083	0.8308	0.8583	0.0155
9	mpgs	0.9000	0.9140	0.9300	0.0107	0.7200	0.7600	0.8000	0.0291	0.8600	0.8880	0.9300	0.0244
10	musk1	0.9913	0.9952	0.9978	0.0022	0.9283	0.9380	0.9500	0.0071	0.5826	0.6948	0.7696	0.0581
11	musk2	1.0000	1.0000	1.0000	0.0000	0.9819	0.9834	0.9844	0.0009	0.8467	0.8544	0.8774	0.0108
12	spam	0.9351	0.9372	0.9392	0.0015	0.9070	0.9095	0.9115	0.0014	0.9338	0.9402	0.9453	0.0033
13	sphe	0.8000	0.8254	0.8538	0.0167	0.7846	0.8050	0.8192	0.0106	0.7846	0.8108	0.8385	0.0154
14	sphf	0.8588	0.8841	0.9000	0.0135	0.8324	0.8479	0.8559	0.0088	0.7941	0.8171	0.8353	0.0137
15	pbri1m	0.8571	0.8571	0.8571	0.0000	0.9000	0.9257	0.9429	0.0148	0.8857	0.9000	0.9286	0.0151
16	pbri1s	0.7667	0.8100	0.8333	0.0211	0.8000	0.8417	0.8667	0.0212	0.7500	0.7883	0.8333	0.0315
17	pbri1r	0.6429	0.6771	0.7286	0.0271	0.6857	0.7043	0.7571	0.0224	0.6429	0.6814	0.7286	0.0278
18	pbri2m	0.8571	0.8571	0.8571	0.0000	0.9286	0.9300	0.9429	0.0045	0.8429	0.8943	0.9286	0.0245
19	pbri2s	0.7667	0.7983	0.8333	0.0183	0.8000	0.8367	0.8667	0.0205	0.7000	0.7500	0.8000	0.0430
20	pbri2r	0.6286	0.7014	0.7429	0.0389	0.6714	0.6914	0.7143	0.0120	0.6000	0.6557	0.7000	0.0333
21	hooc	0.6542	0.6604	0.6708	0.0049	0.6596	0.6596	0.6596	0.0000	0.5875	0.6200	0.6542	0.0229
22	haro	0.8467	0.8660	0.8867	0.0131	0.6071	0.6071	0.6071	0.0000	0.7067	0.7587	0.8267	0.0372
23	iris	0.9533	0.9640	0.9733	0.0056	0.9467	0.9593	0.9667	0.0073	0.9333	0.9520	0.9667	0.0103
24	msgs	0.9864	0.9881	0.9899	0.0011	0.7962	0.8028	0.8088	0.0033	0.9893	0.9912	0.9940	0.0016
25	popa	0.7250	0.7537	0.7750	0.0145	0.6750	0.7112	0.7375	0.0232	0.5250	0.6188	0.6500	0.0383
26	blcl	0.9693	0.9701	0.9708	0.0005	0.9599	0.9616	0.9632	0.0010	0.9721	0.9733	0.9743	0.0008
27	pbri1y	0.5857	0.6186	0.6429	0.0191	0.6000	0.6071	0.6143	0.0075	0.5857	0.6443	0.7286	0.0474
28	pbri2y	0.5571	0.5886	0.6143	0.0200	0.5857	0.5957	0.6143	0.0096	0.5857	0.6343	0.6857	0.0310
29	derm	0.9676	0.9759	0.9824	0.0041	0.9647	0.9682	0.9765	0.0043	0.9618	0.9729	0.9824	0.0068
30	flag	0.5000	0.5294	0.5687	0.0247	0.6167	0.6358	0.6667	0.0171	0.5000	0.5475	0.5750	0.0277
31	caar	0.7722	0.7778	0.7861	0.0056	0.6611	0.6703	0.6750	0.0039	0.6222	0.6475	0.6750	0.0165
Average		0.8183	**0.8369**	0.8533	0.0113	0.7937	**0.8080**	0.8221	0.0095	0.7700	**0.8031**	0.8351	0.0215

Appendix B

RM-RLS Algorithm Implementation in Matlab Code

```
function [F,K] = RMmodel(x,r)
[m,l] = size(x);
K = 1+r+l*(2*r-1);
F = zeros(m, K);
F(:,1) = ones(m,1);
for k=1:r,
    F(:,(k-1)*l+2:k*l+1)=x.^k;
end
for j=1:r,
    F(:,r*l+1+j)=sum(x,2).^j;
end
for j=2:r,
    F(:,r*(l+1)+2+(j-2)*l:r*(l+1)+1+(j-1)*l)=...
         x.*((sum(x,2).^(j-1))*ones(1,l));
end

function [alpha,invM]=e2_RLStrain(old_alpha,old_invM,x,y,r,b,a)
a11=1/(1-a);
a1=1/a;
[F,K]=RM3model(x,r);
alpha=old_alpha;
invM=old_invM;
for k=1:size(F,1)
    invM=invM*a11-a11*a11*invM*F(k,:)'*F(k,:)...*
         invM/(a11*F(k,:)*invM*F(k,:)'+a1);
    invM=(a*invM+eye(K))\invM;
    alpha=alpha+a*invM*F(k,:)'*(y(k)-F(k,:)*alpha);
end
```